الجمعية المصرية لهندسة اللغة

THE EGYPTIAN SOCIETY OF LANGUAGE ENGINEERING

The Tenth Conference On Language Engineering

December 15-16, 2010, Cairo, Egypt

(ESOLEC'2010)

Organized by

Egyptian Society of Language Engineering (ESOLE)

Under the Auspices of

PROF. DR. HANY HELAL
Minister of Higher Education and Scientific Research

PROF. DR. TAREK KAMEL
Minister of Communications and Information

PROF. DR. AHMED ZAKI BADR
Minister of Education

PROF. DR. MOHAMMAD MAGED ELDIEB
President of Ain Shams University

PROF.DR. ALI SHERIF ELFAYAD
Dean, Faculty of Engineering, Ain Shams University

CONFERENCE CHAIRPERSON
PROF. DR. M. A. R. GHONAIMY

CONFERENCE COCHAIRPERSON
PROF. DR. SALWA ELRAMLY

Faculty of Engineering –Ain Shams University

http: //eng.asu.edu.eg/esole

Conference Chairman: **Conference Sponsors**

Prof. Dr. M. R. A. Ghonaimy

Technical Program Committee:

Prof. Taghrid Anber , **Egypt**
Prof. I. Abdel Ghaffar , **Egypt**
Prof. M. Ghaly, **Egypt**
Prof. M. Z. Abdel Mageed, **Egypt**
Prof. Khalid Choukri, ELDA, **France**
Prof. Nadia Hegazy, **Egypt**
Prof. Christopher Ciri, LDC, **U.S.A**
Prof. Mona T. Diab, Stanford U., **U.S.A**
Prof. Ayman ElDossouki, **Egypt**
Prof. Afaf AbdelFattah, **Egypt**
Prof. Y. ElGamal**, Egypt**
Prof. M. Elhamalaway**, Egypt**
Prof. S. Elramly, **Egypt**
Prof. H. Elshishiny**, Egypt**
Prof. A. A. Fahmy**, Egypt**
Prof. I. Farag, **Egypt**
Prof. Magdi Fikry, **Egypt**
Prof. Wafa Kamel, **Egypt**
Prof. S. Krauwer, **Netherlands**
Prof. Bente Maegaard, CST, **Denmark**
Prof. A. H. Moussa, **Egypt**
Prof. M. Nagy**, Egypt**
Prof. A. Rafae, **Egypt**
Prof. Mohsen Rashwan**, Egypt**
Prof. H.I. Shaheen**, Egypt**
Prof. S.I. Shaheen**, Egypt**
Prof. Hassanin M. AL-Barhamtoshy**, Egypt**
Prof. M. F. Tolba, **Egypt**
Dr. Tarik F. Himdi**, Saudi Arabia**

Organizing Committee

Prof. I. Farag	Prof. S. Elramly
Prof. Nadia Hegazy	Prof. Hany Kamal
Prof. H. Shahein	Dr. A. Bahaa
Eng. Mona Zakaria	Eng. Bassant A. Hamid

Conference Secretary General

Prof. Dr. Salwa Elramly

The Tenth Conference on Language Engineering
Final Program

Wednesday 15 December 2010

9.00 - 10.00 Registration

10.00 - 10.30 Opening Session

10.30 - 11.15 **Session 1**: **Invited Paper 1**:
Chairman: Prof. Dr. Adeeb Riad Ghonaimy
Arabic Linguistics and In-Depth Text Processing
Nabil Aly
Expert in Computational Linguistics

11.15 - 12.00 Coffee break

12.00 - 13.30 **Session 2: Natural Language Processing for Information Retrieval**
Chairman: Prof. Dr. Ibrahim Farag

1. **Invited Paper 2: An NLP Based Fully Distributed Arabic Search Engine – Part (1)**
   Taghride Anbar
   Al-Alson Faculty, Ain Shams University
2. **Invited Paper 2: An NLP Based Fully Distributed Arabic Search Engine – Part (2)**
   Mohammad Abdeen
   Faculty of Computer and Information Sciences, Ain Shams University
3. **A Comparative Study of Rocchio Classifier Applied to supervised WSD Using Arabic Lexical Samples**
   Soha M. Eid[1], Almoataz B. Al-Said[3], Nayer M. Wanas[1], Mohsen A. Rashwan[2], Nadia H. Hegazy[1]
   [1]Informatics Department, Electronics Research Institute, Cairo, Egypt
   [2]Electronics and Electrical Communications Department, Faculty of Engineering, Cairo University, Cairo, Egypt
   [3]Cairo University, Cairo, Egypt

13.30 - 14.30 **Session 3: Machine Translation:**
Chairman: Prof. Dr. Mohamad Zaki Abdel Mageed

1. **UNL^{+3}: The Gateway to a Fully Operational UNL System**
   Sameh Alansary[1], Magdy Nagi[2], Noha Adly[2]
   [1]Department of Phonetics and Linguistics, Faculty of Arts, University of Alexandria, El Shatby, Alexandria, Egypt
   [2]Computer and Engineering Department, Faculty of Engineering, University of Alexandria, El Hadara, Alexandria, Egypt
2. **A Practical Application of the UNL^{+3} Program on the Arabic Language**
   Sameh Alansary
   Phonetics and Linguistics Department, Faculty of Arts, University of Alexandria, Alexandria, Egypt

14.30 - 15.30 Lunch

15.30 - 17.00 **Session 4: Room A: Language Analysis and Comprehension:**
Chairman: Prof. Dr. Taghride Anbar

1. **A Machine Learning-Based Automatic Arabic Diacritizer, Tokenizer and Morphological Analyzer**
   Ramy N. Eskander[1], Amin F. Shoukry[2], Saleh A.S. El-Shehaby[3]

[1]*Faculty of Engineering, Alexandria University, Alexandria, Egypt*
[2]*Egypt-Japan University of Science and Technology (EJUST), New Borg El-Arab City, Alexandria, Egypt*
[3]*Medical Research Institute, Alexandria University, Alexandria, Egypt*

2.تحليل للجملة العربية مبني على مدونة نصية من النصوص المعاصرة (MSAC) مساك
د/سلوى السيد حمادة
معهد بُحوث الإلكترونيـّـَات

3. Improving YamCha Tool Performance
Salwa Hamada
Electronics Research Institute (ERI)

15.30 - 17.00 <u>Session 5</u>**: Room B**: **Semantic Web and Ontology Languages:**
Chairman : Prof. Dr. Hassanin El-Barhamtoushy

1. Ontology-based Architecture for an Arabic Semantic Search Engine
Ibrahim Fathy Moawad[1], Mohammad Abdeen[2], Mostafa Mahmoud Aref[3]
[1]*Computer Science Department, Faculty of Computer Science and Information Sciences, Ain Shams University, Cairo, Egypt*
[2]*Information System Department, Faculty of Computer Science and Information Sciences, Ain Shams University, Cairo, Egypt*

2. Rich Semantic Graph Generation System Prototype
Mostafa Mahmoud Aref[1], Ibrahim Fathy Moawad[2], Soha Said Ibrahim[1]
[1]*Computer Science Department, Faculty of Computer Science and Information Sciences, Ain Shams University, Cairo, Egypt*
[2]*Information System Department, Faculty of Computer Science and Information Sciences, Ain Shams University, Cairo, Egypt*

3. Ontology and its Methodology
Susan Fisal Ellakwah[1], Passent El-Kafrawy[2], Mohamed Amin[2], El-Sayed El-Azhary[1]
[1]*Central Lab for Agricultural Expert Systems (CLAES), Agricultural Research Center (ARC), Giza, Egypt*
[2]*Mathematics and CS Department, Faculty of Science, Menoufia University, Egypt*

17.00 - 17.30 <u>Session 6</u>**: Room B**: **Language Engineering and Artificial Intelligence**
Chairman: Prof. Dr. Hany Kamal Mahdy

1. AL-IMAM: A Comprehensive Database for Arabic Text Mining
Ibrahim F. Imam, Ahmed Abd-Allah
Computer Science Department, Arab Academy for Science, Technology and Maritime Transport, Cairo, Egypt

Thursday 16 December 2010

10.00 - 12.00 **Session 7: Room A: Speech Processing, Recognition and Synthesis**

Chairman: Prof. Dr. Mohsen Rashwan

1. **On the Modeling of Non-Keyword Intervals for Spoken-Term Detection in Arabic**
   M. Hesham[1], M. F. Abu-EL-Yazeed[2], and A. Toulan[2]
   [1]Engineering Math. & Physics Dept., Faculty of Engineering, Cairo University
   [2]Electronics & Communications Engineering Dept., Faculty of Engineering, Cairo University, Egypt
2. **Designing and Implementing Arabic Text-To-Speech (ArTTS)**
   Hassanin M. Al-Barhamtoshy, Fahd Al-Hiedary, Mansour Al-Johany and Wajdi H. Al-Jedaibi
   Faculty of Computing and Information Technology, King Abdulaziz University, SA
3. **Automatic Speech Segmentation Using Genetic Algorithm Based on Best Tree Encoding**
   Amr M. Gody
   Electrical Engineering Department, Faculty of Engineering, Fayoum University, Fayoum, Egypt
4. **Variable Bit Rate Speech Coding Using Wavelet Transform**
   Sarah Baligh Boulos[1], Dr. Naglaa Hosny[2]
   [1]Communications and Electronics Department, Faculty of Engineering, Ain Shams University
   [2]Canadian International College, Egypt

12.00 - 12.30 Coffee Break

12.30 - 13.30 **Session 8: Room A: Automatic Optical Character Recognition**

Chairman: Prof. Dr. Waleed Fakhr

1. **Recent Advances in Arabic Handwriting Recognition**
   Mostafa G. Mostafa[1], Mohamed F. Tolba[2]
   [1]Computer Science Department, Faculty of Computer & Information Sciences, Ain Shams University, Cairo, Egypt.
   [2]Scientific Computing Department, Faculty of Computer & Information Sciences, Ain Shams University, Cairo, Egypt
2. **Arabic Character Recognition Using statistical Moment Invariants and ANN**
   Ismail I. Amr[1], Mohamed Amin[2], Passent El-Kafrawy[2], and Amr M. Sauber[2]
   [1]College of Computers and Informatics, Misr International University, Cairo, Egypt
   [2]Mathematics and CS Department, Faculty of Science, Menoufiya University, Egypt

Session 9: Room A: Large Corpora
Chairman: Prof. Dr. Mohamad Fahmy Tolba

13.30 - 14.00 **1. A General Purpose Large Scale Arabic Online Handwriting Corpus**
Sherif Abdou[1,3], Mohamed Waleed Fakhr[2,3], Ibrahim Hosney[1], FakhrEdeen Alwajeeh[1]
[1]Faculty of Computers and Information Cairo University, Egypt
[2]The Arabic Academy of Science and Technology, Egypt
[3]The Arabic Language technologies Center (ALTEC), Egypt

2. Printed-Arabic Large Text Corpus for OCR Research (P-ALTEC)
Waleed Fakhr[1], Mohsen Moftah[1], Mohsen Rashwan[2], Mohamed ElMahallawy[1]
[1]College of Computing, Arab Academy for Science and Technology, Cairo, Egypt
[2]Communications Department, College of Engineering, Cairo University, Giza, Egypt

14.00 - 15.00 Lunch

15.00 - 17.00 **Session 10: Room A: AOCR Products Evaluation Workshop**
Chairman: Prof. Dr. M. Fahmy Tolba

1. المعايير اللغوية والفنية لتقييم برامج التعرف الضوئي على الحروف العربية
عمرو جمعة عبد الرسول
كلية دار العلوم – جامعة القاهرة

2. Panel Discussion

10.00 - 12.00 **Session 11: Room B: Workshop on Language Modeling Applications in Arabic NLP**
Part 1: Prof. Dr. Mohamed Waleed Fakhre

12.30 - 14.00 **Session 12: Room B: Workshop on Language Modeling Applications in Arabic NLP**
Part 2: Eng. Mohsen Moftah

15.00 - 17.00 **Session 13: Room B: Workshop on Language Modeling Applications in Arabic NLP**
Part 3: Eng. Shady Abdel Ghaffar

17.00 - 17:30 Closing session

الجمعية المصرية لهندسة اللغة

THE EGYPTIAN SOCIETY OF LANGUAGE ENGINEERING

أعضاء الجمعية من المؤسسات

1- مركز نظم المعلومات – كلية الهندسة – جامعة عين شمس
2- معهد الدراسات والبحوث الإحصائية – جامعة القاهرة
3- مركز الحساب العلمى – جامعة عين شمس
4- الأكاديمية العربية للعلوم والتكنولوجيا والنقل البحرى
5- أكاديمية أخبار اليوم
6- معهد بحوث الإلكترونيات
7- معهد تكنولوجيا المعلومات
8- مكتبة الإسكندرية
9- المعهد القومى للاتصالات (NTI)
10- الشركة الهندسية لتطوير نظم الحاسبات (RDI)
11- الهيئة القومية للاستشعار من بعد و علوم الفضاء
12- كلية الحاسبات و المعلومات جامعة قناة السويس
13- دار التأصيل للبحث و الترجمة

أهداف الجمعية

1- الاهتمام بمجال هندسة اللغويات مع التركيز على اللغة العربية بصفتها لغتنا القومية والتركيز على قواعد البيانات المعجمية وصرفها ونحوها ودلالتها بهدف الوصول إلى أنظمة ألية لترجمة النصوص من اللغات الأجنبية إلى اللغة العربية والعكس, وكذلك معالجة اللغة المنطوقة والتعرف عليها وتوليدها, ومعالجة الأنماط مع التركيز على اللغة المكتوبة بهدف إدخالها إلى الأجهزة الرقمية.
2- متابعة التطور فى العلوم والمجالات المختصة بهندسة اللغة
3- التعاون مع الجمعيات العلمية المماثلة على المستوى المحلى والقومى والعالمى.
4- إنشاء قواعد بيانات عن البحوث التى سبق نشرها والنتائج التى تم التوصل إليها فى مجال هندسة اللغة بالإضافة إلى المراجع التى يمكن الرجوع إليها سواء فى اللغة العربية أو اللغات الأخرى.
5- إنشاء مجلة علمية دورية للجمعية ذات مستوى عال لنشر البحوث الخاصة بهندسة اللغة وكذلك بعض النشرات الدورية الإعلامية الأخرى بعد موافقة الجهات المختصة.
6- عقد ندوات لرفع الوعى فى مجال هندسة اللغة
7- تنظيم دورات تدريبية يستعان فيها بالمتخصصين وتتاح لكل من يهمه الموضوع. وذلك من أجل تحسين أداء المشتغلين فى البحث لخلق لغة مشتركة للتفاهم بين الأعضاء
8- إنشاء مكتبة تتاح للمهتمين بالموضوع تشمل المراجع وأدوات البحث من برامج وخلافه.
9- خلق مجال للتعاون وتبادل المعلومات وذلك عن طريق تهيئة الفرصة لعمل بحوث مشتركة بين المشتغلين فى نفس الموضوعات.
10- تقييم المنتجات التجارية أو البحثية والتى تتعرض لعملية ميكنة اللغة.
11- رصد الجوائز التشجيعية للجهود المتميزة فى مجالات هندسة اللغة.
12- إنشاء فروع للجمعية فى المحافظات.

الجمعية المصرية لهندسة اللغة

THE EGYPTIAN SOCIETY
OF LANGUAGE ENGINEERING

المؤتمر العاشر لهندسة اللغة

15–16 ديسمبر 2010

القاهرة– جمهورية مصر العربية

ينظم المؤتمر

الجمعية المصرية لهندسة اللغة

تحت رعاية

الأستاذ الدكتور/هانى هلال	**الأستاذ الدكتور/ طارق كامل**	**الأستاذ الدكتور/ أحمد زكى بدر**
وزير التعليم العالى والبحث العلمى	**وزير الاتصالات والمعلومات**	**وزير التربية والتعليم**

الأستاذ الدكتور/ محمد ماجد الديب

رئيس جامعة عين شمس

الأستاذ الدكتور/علي شريف الفياض

عميد كلية الهندسة – جامعة عين شمس

رئيس المؤتمر

الأستاذ الدكتور/ محمد أديب رياض غنيمى

كلية الهندسة – جامعة عين شمس

مقرر المؤتمر

الأستاذ الدكتور / سلوى حسين الرملى

كلية الهندسة – جامعة عين شمس

مكان عقد المؤتمر : كلية الهندسة – جامعة عين شمس

http: //eng.asu.edu.eg/esole

Table of Contents

University, Fayoum, Egypt

A Machine Learning-Based Automatic Arabic Diacritizer, Tokenizer and Morphological Analyzer

Ramy N. Eskander[*], Amin F. Shoukry[**], Saleh A.S. El-Shehaby[***]

[*]*Faculty of Engineering, Alexandria University*
Alexandria, Egypt
rami_iskander@hotmail.com

[**]*Egypt-Japan University of Science and Technology (EJUST)*
P.O.Box 179, New Borg El-Arab City, Alexandria, Egypt
amin.shoukry@ejust.edu.eg

[***]*Medical Research Institute, Alexandria University*
Alexandria, Egypt
sshehaby@gmail.com

***Abstract*— Arabic language is a language that does not lend itself, easily, to automatic processing. This is caused by many issues such as the affixation system in Arabic, the omission of disambiguating short vowels, and the diacritization system which may be absent from the text. Moreover, Arabic letters are usually written in different shapes according to their positions in a word. Accordingly, an Arabic word has about ten possible morphological solutions (analyses); with only one being the correct solution; depending on the context of the word. Learning is required to understand context.**

In this study, a machine-learning-based Arabic automatic tagger is implemented. The tagger acts as an automatic diacritizer, tokenizer and morphological analyzer. The tagger is trained on a corpus whose Buckwalter morphological analysis is known and depends on a set of SVMs (Support Vector Machines) for classification. Each classifier is trained, separately, on a single morphological feature out of 13 features; which represent an extension of the set (of 10 features) provided by the Buckwalter analyser. This divide-and-conquer technique has proved to be efficient. A best-match technique is used to pick the correct Buckwalter morphological solution consistent with the output provided by the trained classifiers. The tagging results are promising and competitive relative to existing systems. The authors plan to extend it to domain specific Arabic language understanding tasks.

1 INTRODUCTION

The Arabic language is one that is difficult to handle automatically. This is due to the following issues:

- the existence of affixation (prefixes or suffixes) with the Arabic words,
- the omission of disambiguating short vowels,
- the diacritization system which may be absent from the text,
- the different variants in which some Arabic letters may be written.

As a result, most of the Arabic words are ambiguous in their morphological analysis. Generally, for an Arabic word there are about ten possible morphological solutions (analyses) on average; with only one being the correct analysis according to the context of the word. For instance; the Arabic word 'وجد' may have the meanings of 'passion', 'and + grandpa', 'and + seriousness, and 'found', with the following possibilities for the noun forms: nominative, accusative or genitive, and the verbal forms: perfect or imperative.

In order to extract the correct solution of an Arabic word, it is a must to look at the context of this word. While humans understand the meaning of the context first, machines need some kind of learning, since no direct rules can resolve the morphological analysis of any natural language.

An Arabic word consists of 3 parts; 0-3 prefixes, 1 stem and 0-3 suffixes, where the prefixes and suffixes are called the affixes of the word. Prefixes always have a length of 0-4 characters, while suffixes always have a length of 0-6 characters. A stem can have any number of characters, but should have at least 1 character.

The objective of this study is to make an Arabic automatic tagger that acts as a diacritizer, tokenizer and morphological analyzer. This is to be done through using a strong Arabic morphological analyzer. The Buckwalter morphological analyzer [2] is the one that is recommended for this purpose due to previous related work and good results. The Buckwalter analyzer accepts an Arabic text as an input and outputs all the possible morphological solutions for each word in the input text. A complete Buckwalter solution contains:

- the complete POS of the word; the POS includes word diacritization, word tokenization and all the morphological features of the word, such as basic POS, gender... etc.,
- the English gloss of the word, the English translation of the word (e.g., ktba = wrote + [he/it]),
- word vocalization (diacritization), the pronunciation of the word,
- word lemma, a distinct form of the word that has similar meanings (e.g., mdrsp = school or teacher).

These solutions correspond to the values of a large number of features, such as the basic Part-Of-Speech (POS), voice, gender, person… etc. For Arabic, this gives us about 300,000 theoretically possible completely specified Buckwalter morphological POS's.

In order to train classifiers on this huge set (300,000 possible complete POS's), a divide and conquer technique seems to be a good solution. The set of POSs is split into smaller sets, and a classifier is trained for each set independently. SVMs have been chosen for this purpose. When testing the classifiers, their outputs are combined together in order to reach the correct POS and select the correct Buckwalter analysis.

The remaining of this paper is organized as follows: section 2 illustrates how the presented work relates to previous known work. Section 3 gives a general overview of the proposed system. Section 4 describes the data preparation phase while section 5 describes the classifiers training and testing. Section 6 describes the obtained results and, finally, section 7 concludes the paper.

2 RELATION TO PREVIOUS WORK

While there have been many publications on computational morphological analysis for Arabic, only Diab et al. [7] and Habash et al. [8] performed a large-scale corpus-based (supervised) evaluation of their approach. They mainly depend on the Buckwalter morphological analyzer, and use SVM-based learner for their automatic tagging tasks. Although the system presented, in this work, is similar to the one used by Habash et al. [8], the **main differences** are:

- Habash et al. specifies a set of ten morphological features, for each word, within an input Arabic text. These features are the basic POS: conjunction, particle, pronoun, determiner, gender, number, person, voice and aspect. These features are not enough to specify the complete Buckwalter solution. Therefore, three more features have been added, which are indefiniteness, case and mood.
- The target of the tagger is to make an automatic tagger that acts as a diacritizer, tokenizer and morphological analyzer. This is to be done by automatically assigning the complete Buckwalter analysis and not just a set of morphological features. By knowing the correct Buckwalter analysis, much valuable information can be obtained such as the complete POS of the word, the English translation of the word, word vocalization (diacritization) and word lemma.
- Selecting the possibly correct Buckwalter analysis is done through an advanced best-match algorithm designed to deal with morphological data and makes use of the history of previous tagging results.
- Many issues have been investigated, such as the alternatives and the different parameters values that affect the tagging process such as the selection of the features and their possible values, and the tuning of the learning parameters while training the classifiers.

3 OVERVIEW OF THE PROPOSED SYSTEM

The development of the proposed system necessitates three phases. The first phase is the construction of the training and testing data sets from an Arabic Corpus. The training data is constructed by converting an annotated Arabic corpus to what is called “Morph objects” through a “Morph generator” (implemented according to the domain of the features and their values). The second phase is the supervised training of the SVM-based classifiers [6] using the training data set. The third phase consists in applying the trained classifiers on the untagged input Arabic text, and processing the output of these classifiers to get the possibly correct Buckwalter analysis. By comparing the obtained results with the known correct Buckwalter solutions, the performance of the system is evaluated.

4 DATA PREPARATION

The source of the data (training corpus) is the Penn Arabic Treebank Corpus (PAT), developed by the Linguistic Data Consortium (LDC). This corpus has been used in the previous Arabic automatic taggers developed by Diab et al [7] and Habash et al [8], and proved to be an excellent source when it comes to Buckwalter-based data.

The PAT corpus is a huge collection of Arabic text, generated from different sources at different times and contexts. Each word in the PAT corpus is associated with all the possible Buckwalter solutions, with the correct one identified. The PAT corpus is not well-written which means that it may contain Arabic text that does not follow strict syntactical or grammatical rules. For each word in the PAT Corpus, the corpus lists all the solutions corresponding to the possible different variants of that word, e.g., it tries to replace the Arabic letter ALEF with ALEF-with-HAMZA-Above and ALEF-with-HAMZA-Below plus the other possible variants for the Arabic letter ALEF. The same is done with the Arabic letter HEH and THE-MARBUTA, and the Arabic letter ALEF-MAKSURA and YEH. Also, the corpus lists all the solutions corresponding to the possible diacritization marks of the word. Additionally, the analysis of a word always contains a proper noun solution whose gloss in not in the lexicon.

The PAT corpora have three main parts. For this study, part 3 (with a size of 340,000 words) has been used. It has been initially chosen to use 327k words for the training purpose while the remaining 13k words are used for testing and analyzing the developed system. In general, increasing the size of the training corpus enhances the tagger performance. But at some point, increasing the size of the corpus will not affect the final output of the tagger, which is known as a 'Saturation State". This happens when the classifier is not capable of generating any further information from any extra training data, i.e., the information within the extra training data is redundant.

Each word in the training corpus is converted into a Morph object by splitting the correct POS of this word into a group of morphological features. This process is not easy in several cases, and many decisions have to be determined. This conversion process is done by what is called a 'Morph generator'. A Morph generator has as input a Buckwalter solution and outputs its corresponding Morph object. As mentioned before, a word may have several Buckwalter solutions with one of them being the correct one according to the context of the word. A Buckwalter solution includes the following information; the complete POS, English gloss, vocalization and lemma.

A Morph object represents all the morphological features of a word based on a specific Buckwalter solution. For a given word, if there are 'n' different Buckwalter solutions, then there are 'n' possible Morph objects, each corresponding to a specific Buckwalter solution.

In order to implement the Morph generator Application, it is required to:

1. Determine the morphological features that a Morph object should contain, the selection of the features should be sufficient to identify a unique complete Buckwalter POS out of different possible complete Buckwalter POSs
2. Determine the domain of each morphological feature.
3. Specify rules to obtain the value of each morphological feature from the input Buckwalter solution.

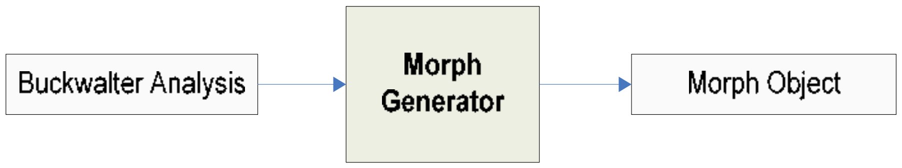


Figure 1: Morph generator

The definition of each of the 13 features used is as follows:

The basic POS of the word, e.g., noun, verb, pronoun, particle…… etc. (نوع الكلمة)
The existence of a determiner associated with the word (التعريف بال)
The existence of a particle associated with the word (وجود حرف متصل)
The existence of a pronoun associated with the word (وجود ضمير متصل)
The existence of a conjunction associated with the word (وجود حرف عطف)
The gender of the word (نوع الجنس)
The number of the word (العدد)
The person of the word (الشخص)
The voice of the word (only with verbs) (بناء الفعل)
The aspect of the word (only with verbs) (زمن الفعل)
The indefiniteness of the word (الإضافة)
The mood of the word (only with verbs - like the case in the nouns) (حالة الفعل)
The case of the word (nominative -accusative - genitive) (الحالة الإعرابية)

The basic POS is the most important feature. It is determined according to the stem of the word. The different types of stems are put in different categories (see Table(1)). The other features depend mainly on the basic POS, i.e., the value of the other features may relate to the value of the basic POS of the word (see Table(2)).

TABLE 1: BASIC POS TYPES

Basic POS	Stem Types
Abbreviation (ABR)	ABBREV
Adjective (ADJ)	ADJ
Adverb (ADV)	ADV
Conjunction (CONJ)	CONJ
Demonstrative Pronoun (DPRO)	DEM DEM_PRON_F DEM_PRON_FD DEM_PRON_FS DEM_PRON_MD DEM_PRON_MP DEM_PRON_MS
Interjection (INTERJ)	INTERJ
Negative Part (NEG_PART)	NEG_PART
Noun (NN)	NOUN
Proper Noun (NNP)	NOUN_PROP FUNC_WORD NUMERIC_COMMA LATIN
Number (NUM)	NUM
Particle (PP)	DET PREP PART EXCEPT_PART FOCUS_PART FUT_PART INTERROG_PART VERB_PART SUB_CONJ REL_ADV
Pronoun (PRON)	PRON_1P PRON_1S PRON_2FS PRON_2MP PRON_2MS PRON_3D PRON_3FP PRON_3FS PRON_3MP PRON_3MS
Punctuation (PUNC)	PUNC
Relative Pronoun (RPRON)	REL_PRON
Verb (VB)	VERB_IMPERATIVE VERB_IMPERFECT VERB_IMPERFECT_PASSIVE VERB_PERFECT VERB_PERFECT_PASSIVE
Unknown (X)	UNKNOWN

NIL	Gloss: not in lexicon

TABLE 2: FEATURE DOMAINS

Feature	Domain
Basic Part Of Speech (POS)	See Table 1
Aspect (ASP)	Perfect(PER) – Imperfect(PER) – Imperative(IMP) – Not Valid(NV)
Case (CASE)	Nominative(NOM) – Accusative(ACC) – Genitive(GEN) – Acc/Gen(AG) – Not Valid(NV) – Not Applicable(NA)
Conjunction (CONJ)	True(T) – False(F) (either exists or not) – Not Applicable(NA)
Determiner (DET)	True(T) – False(F) (either exists or not) – Not Valid(NV)
Gender (GEN)	Masculine(M) – Feminine(F) – Neutral(N) – Not Valid(NV)
Indefiniteness (INDEF)	True(T) – False(F) (either exists or not) – Not Valid(NV)
Mood (MOOD)	Indicative(I) – Subjunctive(S) – Jussive – Subjunctive/Jussive(SJ) – Not Valid(NV) – Not Applicable(NA)
Number (NUM)	Single(SG) – Dual(DL) – Plural(PL) – Not Valid(NV)
Particle (PART)	True(T) – False(F) (either exists or not)
Person (PER)	One(1) – Two(2) – Three(3) – Not Valid(NV)
Pronoun (PRON)	True(T) – False(F) (either exists or not) – Not Valid(NV)
Voice (VOICE)	Active(ACT) – Passive(PAS) – Not Valid(NV)

In order to settle the rules that assign the values of the 13 morphological features it is necessary to determine the following information for each feature:

1- the possible POS type that may carry this feature, i.e., the feature has a 'Not valid' value for the other basic POS types,
2- the default value of the feature in the case of not being able to assign a specific value,
3- the method of extracting the feature value based on the information from (1) and (2).

In order to obtain such information, it is necessary to take into consideration all the possible forms of the Buckwalter POS. Since each Arabic word has 3 main parts; prefixes (0-3) stem (1) and suffixes (0-3), hence, the Buckwalter POS of a word can have 3 POS parts, resulting in more than 300,000 [8] possible POS forms. Therefore, to get the information regarding each feature, it is necessary to iteratively scan all the possible Buckwalter solutions in the training corpus, and studying the relations between the features and their corresponding basic POS types, and observing the value of each feature under all possible cases.

Some features always have a value regardless of the basic POS type of the word, while others may have no-value with specific types; for instance, a conjunction always exists for any basic POS type (either the word is associated with a conjunction or not), while the *aspect* feature is not valid except for the verbs. Therefore, a feature domain may contain an extra value which is "Not Valid". Table (3) lists the possible basic POS types for each feature, and its corresponding default value.

TABLE 3: POSSIBLE BASIC POS TYPES OF THE MORPHOLOGICAL FEATURES

Feature	Possible basic POS Types	Def. Value
Basic POS (POS)	All	X

Aspect (ASP)	VB	Perfect
Case (CASE)	ADJ, ADV, DPRO, INT, NEG, NN, NNP, NUM, PP, RPRO	Nominative
Conjunction (CONJ)	All	False
Determiner (DET)	ABR, ADJ, INT, NEG, NN, NNP, NUM	False
Gender (GEN)	ADJ, ADV, DPRO, NN, NNP, NUM, PRO, RPRO, VB	Masculine
Indefiniteness (INDEF)	ADJ, ADV, DPRO, INT, NEG, NN, NNP, NUM, PP, RPRO	False
Mood (MOOD)	VB	Indicative
Number (NUM)	ADJ, ADV, DPRO, NN, NNP, NUM, PRO, RPRO, VB	Singular
Particle (PART)	All	False
Person (PER)	ADJ, ADV, DPRO, INT, NEG, NUM, NN, NNP, PP, PRO, VB	Three
Pronoun (PRON)	ADJ, ADV, INT, NEG, NN, NNP, NUM, PP, VB	False
Voice (VOICE)	VB	Perfect

From what has been given above, the number of constructed Morph objects is equal to the size of the training corpus. These objects represent the basis for the training data, and are formatted for the training of the classifiers. Figure 2 illustrates the conversion of the training corpus to training Morph objects.

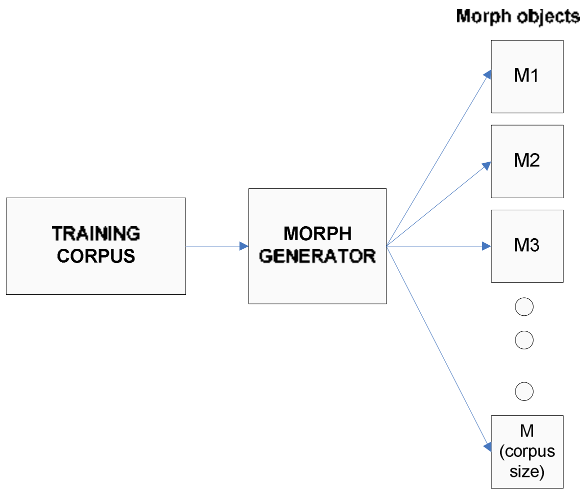


Figure 2: Generating the training Morph objects

5 Classifiers Training and Testing

In the data preparation phase, many thousands of Arabic words are generated; each is associated with its morphological features in the form of a Morph object. The target of this section is to explain the training of the classifiers using these morphological features, so that the classifiers can be tested on any text required for tagging.

Yamcha classifiers, based on Support Vector Machines (SVMs), are used for the training and testing phases. The Morph objects, generated in the data preparation phase, are converted into training files that have the correct format required for Yamcha. Each feature is to be learned independently, i.e., there is an independent training file and an independent classifier for each of the 13 morphological features (see Figure(3)). Each classifier is trained independently resulting in what is called a "Model" file which represents the trained classifier. These model files are then used for testing untagged input data and converting them to output data. Figure 4 illustrates the usage of Yamcha in training the classifiers and outputting the model files. Figure 5 illustrates the testing process.

A feature value of a word in a sentence is most likely correlated to the values of this feature or other features in the adjacent words. For instance, if a word is preceded by a preposition then it is nominative, and when a word has a determiner and is preceded with an accusative word, then it is accusative. Generally speaking, during training, it is necessary to have a look at the grammatical context of the analyzed words. This is DONE in Yamcha through the window Size parameter and the static and dynamic features. The window size parameter determines how many adjacent pieces of information are considered while training the classifiers. All features have been trained under window sizes that have a range between 1 and 5.

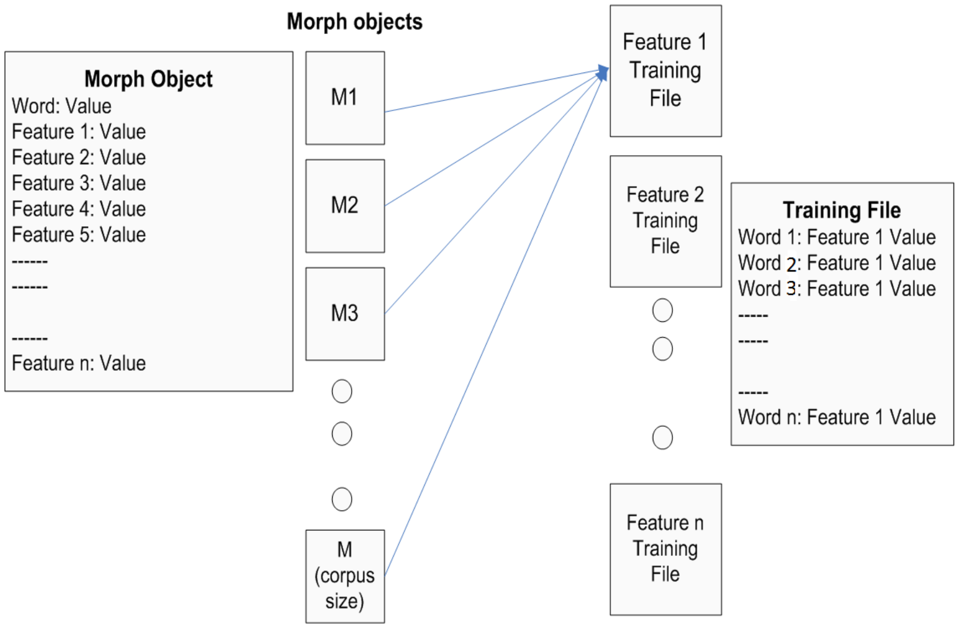


Figure 3: Converting the Morph objects to training files

The learning process consumes much time; the bigger the window size and the learning domain are, the bigger time the learning process consumes. This is not a problem since learning is done only once. However, testing should be fast since it is done frequently, i.e., each time a new untagged input text is queried.

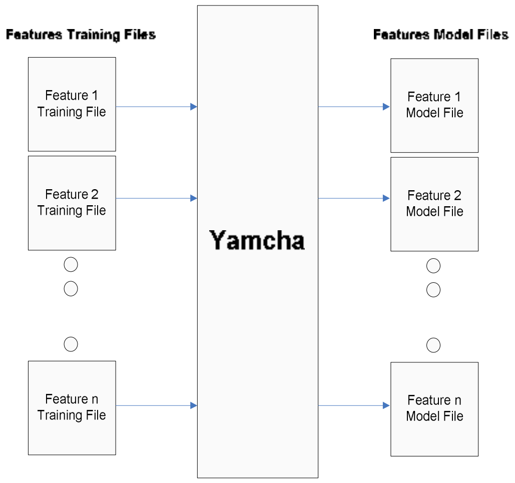


Figure 4: Training the classifiers

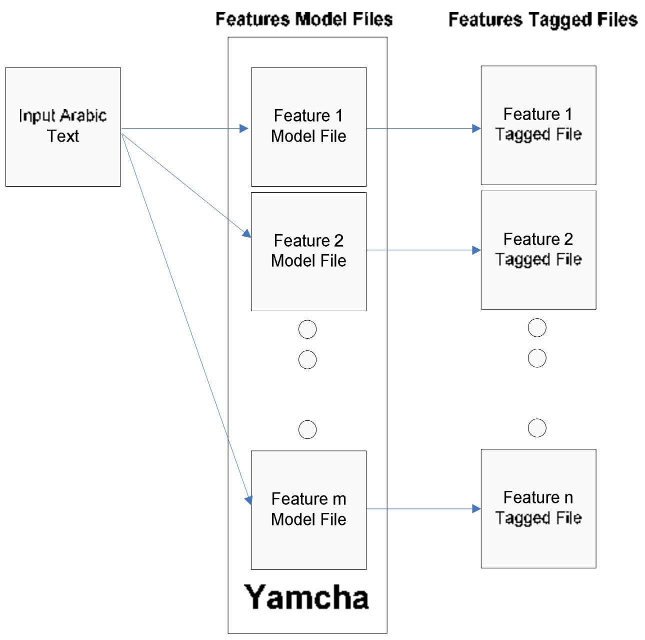


Figure 5: Testing process

The untagged input Arabic text that is required for tagging should have a specific format that is acceptable by Yamcha, where a new line character represents the sentence boundary and, the words including the punctuation marks and any individual tokens, are space-separated.
The suggested Morph objects represent the morphological solutions provided by the classifiers. They are generated by using the 13 tagged files generated in the test phase. These tagged files are then merged together into one tagged file. Figure 6 illustrates the merging process.

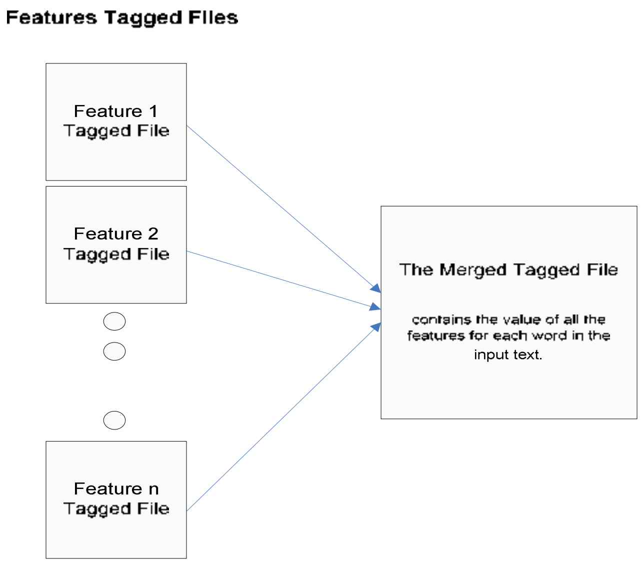


Figure 6: Merging the tagged files

The merged tagged file is then encapsulated into a list of Morph objects using the Morph generator application that was used in generating the training Morph objects. Figure 7 illustrates this process. There is one morph object for each word. This Morph object is the suggested solution provided by the classifiers, and is called the suggested Morph object. The suggested Morph objects are then compared with all the possible Morph objects resulted from the Buckwalter analysis, but first they have to get some kind of tuning.

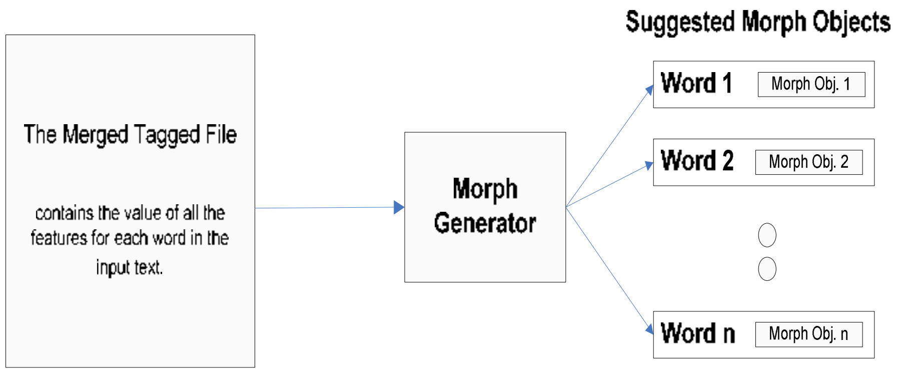


Figure 7: Generating the suggested Morph objects

The aim of the tuning process is to make the output of the classifier more consistent with the Arabic rules. For example, it has been noted that some feature values make conflict with other features values and some feature values do not match the syntax of the underlying word. Therefore, these values are modified in order to be consistent with Arabic morphological rules. Tuning the morphological rules is a linguistic issue and is open for enhancement as there are many rules to be considered.

Next, the suggested Morph objects are compared with the candidate Morph objects obtained by running the Buckwalter analysis on the input untagged text. Figure 8 illustrates the generation process of the candidate Morph object, while Figure 9 represents a full picture of the process of automatically selecting the possibly correct Buckwalter analysis.

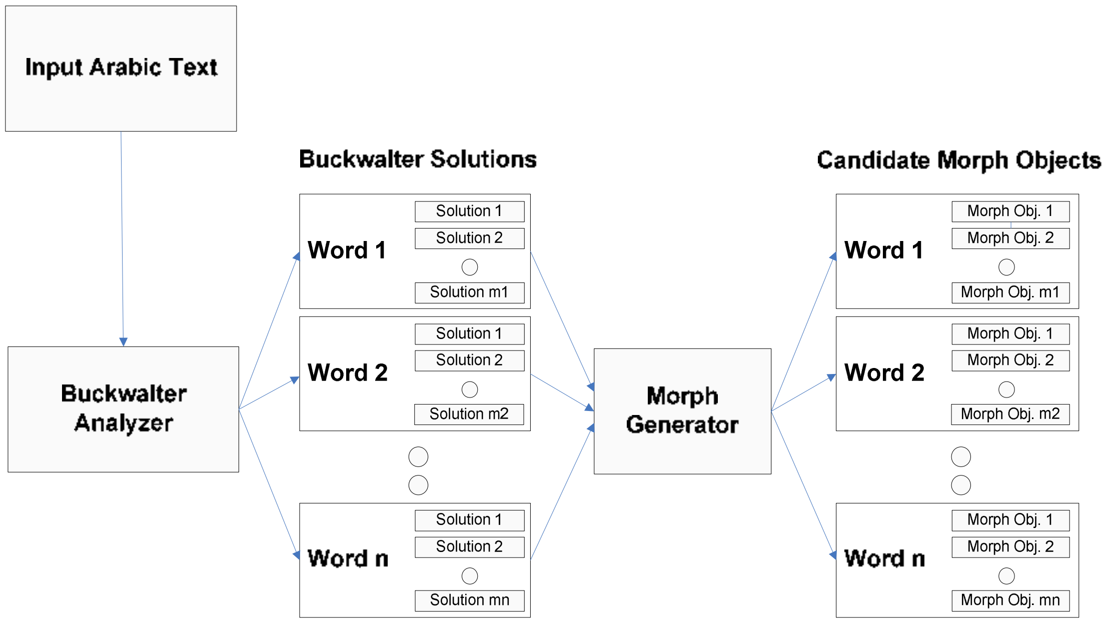


Figure 8: Generating the candidate Morph objects

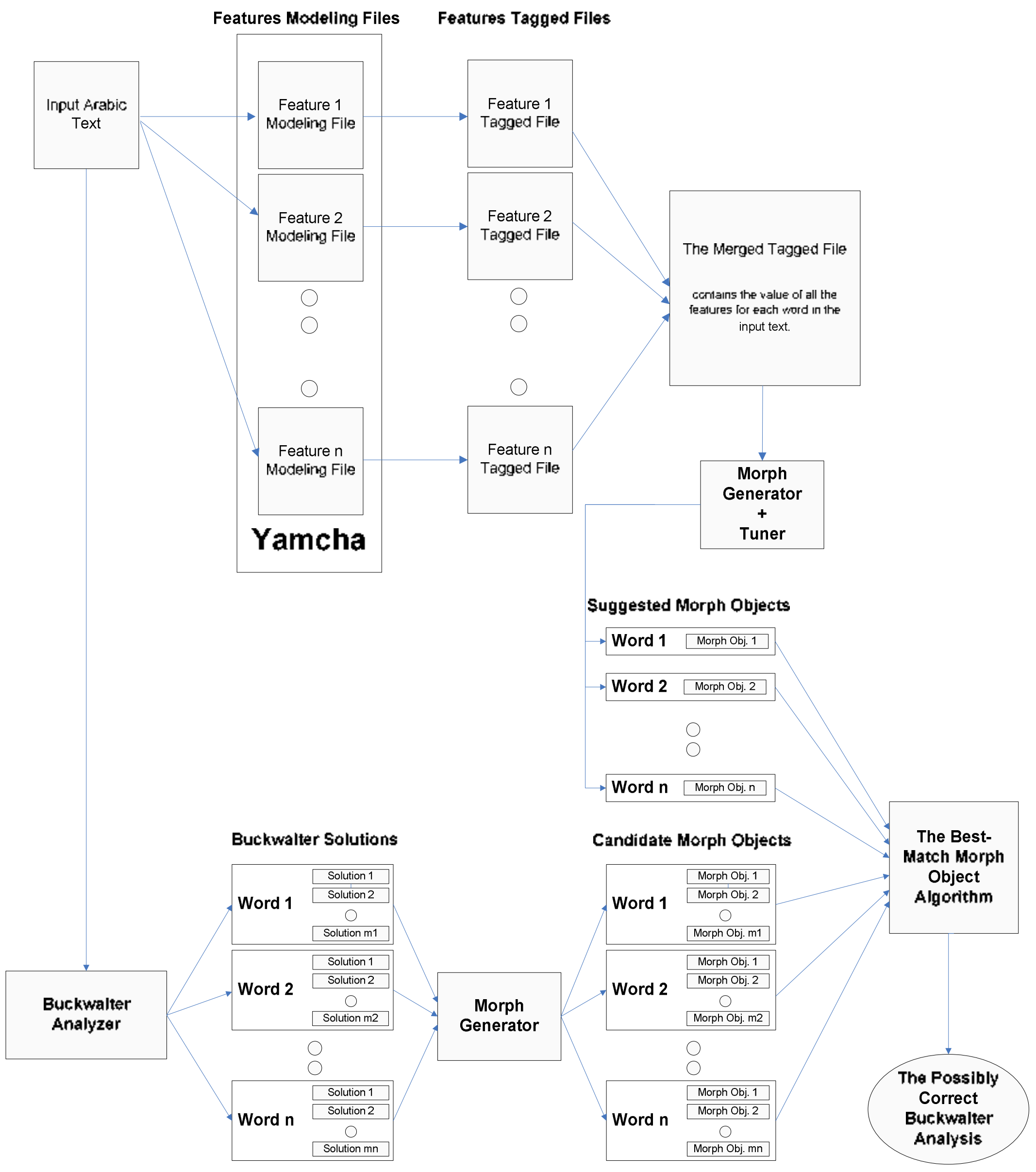

Features Modeling Files
Features Tagged Files
Input Arabic Text
Feature 1 Modeling File
Feature 2 Modeling File
Feature n Modeling File
Yamcha
Feature 1 Tagged File
Feature 2 Tagged File
Feature n Tagged File
The Merged Tagged File
contains the value of all the features for each word in the input text.
Morph Generator + Tuner
Suggested Morph Objects
Word 1 Morph Obj. 1
Word 2 Morph Obj. 2
Word n Morph Obj. n
Buckwalter Solutions
Candidate Morph Objects
Buckwalter Analyzer
Word 1 Solution 1 Solution 2 Solution m1
Word 2 Solution 1 Solution 2 Solution m2
Word n Solution 1 Solution 2 Solution mn
Morph Generator
Word 1 Morph Obj. 1 Morph Obj. 2 Morph Obj. m1
Word 2 Morph Obj. 1 Morph Obj. 2 Morph Obj. m2
Word n Morph Obj. 1 Morph Obj. 2 Morph Obj. mn
The Best-Match Morph Object Algorithm
The Possibly Correct Buckwalter Analysis

Figure 9: Selecting the possibly correct Buckwalter analysis

A The Best-Match Morph Object Algorithm (BMMO)

The purpose of the BMMO algorithm is to find the best-match of a given Morph object with a list of candidate Morph objects. Before discussing the BMMO algorithm, it is necessary to describe some concepts that are used within the algorithm. These concepts are:
feature weight, Morph agreement, awarded and penalized Morph Objects and Morph frequency. They are briefly explained below.

1) *Feature Weight: The weight of a morphological feature describes how much the classifier of this feature is trusted. It is equal to the maximum learning accuracy obtained from training the classifiers.*

2) *Morph Agreement:* Morph agreement is a measure of the similarity between two Morph objects. It is based on the values of the morphological features within the two Morph objects. The agreement is either absolute or weighted. The absolute agreement is the number of the equal features within the two compared Morph objects (an integer in the interval [0, 13]). The weighted agreement is the sum of the weights of the equal features within the two compared Morph objects.

3) *Awarded and Penalized Morph Objects:* When comparing the suggested Morph object provided by the classifiers to the list of the candidate Morph objects provided by the Buckwalter analysis, some candidate Morph objects seem preferable than others. Deciding which Morph objects should be awarded or penalized is a linguistic issue and is open for enhancement as there are many rules to be considered. For example, a Morph object suggesting a noun with no associated case should be penalized as this solution does not make sense. In this work, a positive or negative credit is added to the calculated agreement. The amount of credit to be added has been determined by a trial-and-error technique.

4) *Morph Frequency*: Morph frequency represents how frequent a Buckwalter solution is associated to a Morph object in the training corpus or from previous tagging results. It is a simple type of statistical analysis used in natural language processing. More precisely, Morph frequency is a function of the occurrence of the individual components of the Buckwalter solution in the previous history. These individual components are the POS, English gloss, vocalization and lemma. Mathematically Morph frequency is given as:

$$Morph\ Frequency = (X1.POS + X_2.Gloss + X_3.Voc + X_4.Lemma) / (X_1 + X_2 + X_3 + X_4)$$

Where, *POS*, *Gloss, Voc, and Lemma* represent the frequencies of the occurrence of the POS, English gloss, vocalization and Lemma respectively in the previous historical data. X_1, X_2, X_3 and X_4 are weighting factors that are determined by some kind of trial-and-error technique.

The BMMO algorithm is described as follows:

```
The Best-Match Morph Object Algorithm – BMMO

Input
        A Morph object (suggested Morph object)
        A list of Morph objects (candidate Morph objects)
Output
        The candidate Morph objects list sorted due to the best-match with the suggested Morph object.
```

```
The Best-Match Morph Object Algorithm – BMMO

Procedure

    1-  Calculate the agreement (either absolute or weighted) between each candidate Morph Object and
        the suggested Morph object.

    2-  Add an appropriate credit (either positive or negative) to each agreement value calculated in (1)
        due to the nature of the associated Buckwalter solution.

    3-  Calculate the Morph frequency for each Morph object according to the following equation:
        $Morph\ Frequency = (X1.POS + X_2.Gloss + X_3.Voc + X_4.Lemma) / (X_1 + X_2 + X_3 + X_4)$

    4-  For each candidate Morph object, add the value calculated in (1 & 2) + the value calculated in
        (3) which is equivalent to:
        Agreement + awarded or penalized credit + Morph frequency

    5-  Sort the candidate Morph objects list according to the values calculated in (4).
        In the case of equality, the preference is given to the Morph object that has a higher agreement
```

According to the presented system, the following parameters and choices were considered while applying the best-match algorithm:

- Agreement: The weighted agreement is used instead of the absolute agreement.
- *NOT_IN_LEXICON* solutions are penalized by a negative credit of 3, added to the corresponding agreement.
- Noun without cases solutions are penalized by a negative credit of 1, added to the corresponding agreement.
- Morph Frequency = *(POS + Gloss + Voc + Lemma) / 4*, where X_1, X_2, X_3 and X_4 are all set to 1.

6 RESULTS

There are two variants of the presented system; normal and strict. The normal variant expects the received Arabic input text to be not well written, i.e., includes syntactical errors (e.g., eliminating the letter *HAMZA* while it is required). The normal variant deals with training data generated from not well-written text, and has its own model files and features weights. Also, when applying the best match algorithm, the normal variant considers all the variants generated from the Buckwalter analysis. On the contrary, the strict variant expects the received Arabic input text to be well written, i.e., does not include syntactical errors. This variant deals with training data generated from well-written text, and also has its own model files and features weights. Also, when applying the best match algorithm, the strict variant eliminates the variants that do not match the original tested text, and are generated from the Buckwalter analysis.

Table 4 lists the final obtained accuracies for all the13 morphological features

TABLE 4: FINAL ACCURACIES OF THE MORPHOLOGICAL FEATURES

Morphological Feature	**Accuracy %**	
	Normal Variant	**Strict Variant**
POS	96.7	96.9
Aspect	99.3	99.4
Case	91.3	91.5
Conjunction	99.9	99.9
Determiner	99.1	99.2
Gender	98.7	98.7
Indefiniteness	95.4	95.5
Mood	99.2	99.3
Number	98.6	98.6
Particle	99.8	99.9
Person	99.2	99.2
Pronoun	99.3	99.3
Voice	99.1	99.2

The **aspect**, **mood** and **voice** features have a very high accuracy of about 99.2% because these features appear only with verbs and have an invalid value with the other POS types (verbal features). Because verbs appear less frequently than nouns and particles, the "invalid" value within the verbal features is the major one, and hence the verbal features have high accuracies. Notice that the learning of these features requires a relatively large window size, since the recognition of verbs requires inspecting the surrounding words.

The **conjunction**, **determiner**, **gender**, **number**, **particle**, **person** and **pronoun** features have very high accuracies of 99.9%, 99.2%, 98.7%, 98.6%, 99.9%, 99.2% and 99.3% respectively. These features distinguish their associated words with static and limited prefixes and suffixes, and hence their inspection is not difficult, and their learning has high accuracy. Notice that the learning of these features requires a relatively small window size because the associated words have specific affixes that can distinguish the presence of the inspected features, while a larger window size adds additional misleading information.

The **POS** feature has an accuracy of 96.9% which is much less than the accuracy of the previous features. This is because the POS feature has a large domain consisting of 17 values. Moreover, the POS value depends on the context of the word which in turn makes the learning process more difficult.

The **case** and **indefiniteness** features have low accuracies of 91.5% and 95.5% respectively. This is because these features heavily depend on the location of the word within the sentence, which in turn makes their learning more difficult, and explains why they need a relatively large window size.

It is noticeable that the strict variant has higher accuracies rather than the normal variant. This is because a token in the strict variant has less features values in the training data. Moreover, when applying the best-match algorithm, the selection is done with a less number of candidate Morph objects where the Buckwalter variants that do not match the tested token are eliminated.

Table 5 lists the percentages of the occurrence and the accuracies of the different basic POS types in the PAT testing corpus. Ten basic POS types are detected correctly with an accuracy of higher that 95%. These types are Conjunctions, demonstrative pronouns, interjections, nouns, numbers, particles, pronouns, punctuations, relative pronouns and verbs. Other POS types have also good accuracies, while other types have lower but still-acceptable accuracies.

TABLE 5: FINAL ACCURACIES OF THE BASIC POS TYPES

Basic POS Type	**Occurrence %**	**Accuracy %**	
		Normal Variant	**Strict Variant**
Abbreviation	00.15	70.0	70.0
Adjective	10.70	90.1	91.6
Adverb	00.67	86.3	87.3
Conjunction	00.54	100.0	100.0
Demonstrative pronoun	01.07	100.0	100.0
Interjection	00.05	100.0	100.0
Negation	01.23	93.2	93.8
Not In Lexicon	01.50	79.6	80.2
Noun	36.03	98.0	97.9
Proper noun	05.73	93.5	93.2
Number	01.50	94.9	96
Particle	16.71	99.5	99.5
Pronoun	00.54	100.0	100.0
Punctuation	11.09	100.0	100.0
Relative pronoun	01.66	95.4	95.4
Verb	09.97	96.1	96.4
Unknown	00.89	85.4	86.1

Conjunctions, **demonstrative pronouns**, **interjections**, **pronouns** and **punctuations** are always detected correctly (have an accuracy of 100%). This is because these POS types have limited forms that can be easily recognized in the input text. Punctuations have an extra advantage that they have only one Buckwalter solution, so they are always detected correctly.

Nouns, **numbers**, **particles**, **relative pronouns** and **verbs** have good accuracies of higher than 95%. Their detection accuracies are 97.8%, 96.0%, 99.5%, 95.4%, and 96.4% respectively. It is important to have nouns, verbs and particles with these high accuracies, since they are considered as the basic tokens of any natural language.

Adjectives, **negations** and **proper nouns** have also good accuracies of 91.6%, 93.8% and 93.2% respectively. These accuracies are acceptable since these types rarely occur in the natural language. Moreover, there should be conflicts among nouns, adjectives and proper nouns. This conflict is mostly resolved for nouns, which in turn decreases the detection accuracies of adjectives and proper nouns. This is acceptable since the occurrence frequency of nouns is the highest one among all the basic POS types, and then the detection of nouns is the most important.

Abbreviations, **adverbs**, **not in lexicon types** and **unknown types** have relatively lower accuracies of 70.0%, 87.3%, 80.2% and 86.1% respectively. The detection accuracy of the adverbs should be better than that, while the other types are most likely to be absent in the training data, so their occurrence in any tested text is mostly surprising, and as a result their detection accuracies are the least among the detection accuracies of the other POS types.

Table 6 lists the final overall accuracies of the presented tagger. The following may be observed: Word **vocalization** has an accuracy of 87.7%, which means that diacritization can be assigned with an accuracy of 87.7% on the letter basis. **Word Tokenization** has a very high accuracy of 99.6%. Tokenization is not a separate part of the Buckwalter solution, but it can be extracted from the complete Buckwalter POS, where the words are divided to prefixes, a stem and suffixes. Finally, the tagger can assign a complete Buckwalter solution correctly (POS, gloss, vocalization and lemma ID) with an accuracy of 84.3%. These final results are promising, competitive and high enough to overcome the current lingual systems that deal with the Arabic language.

TABLE 6: FINAL OVERALL ACCURACIES

Solution Part	**Accuracy %**	
	Normal Variant	**Strict Variant**
Complete POS	84.7	85.7
English Gloss	91.3	92.3
Vocalization (Diacritization)	86.7	87.7
Lemma ID	94.5	95.2
Tokenization	99.6	99.6
Complete Buckwalter Solution	83.3	84.3

The tagger outputs 2 main files; the first file includes all the morphological features of the input untagged text, while the second one outputs the input text in a diacritized format. The diacritization is complete including the main case of the word. The following example illustrates the power of the diacritizatoin of the presented system.

Input text:

أكد مصدر مسؤول بالهيئة العامة للرقابة المالية أن الهيئة أوشكت على الانتهاء من إعداد مشروع قانون صناديق التأمين الخاصة والمعاشات الاختيارية.

Text after diacritization:

أَكّدَ مَصْدَرٌ مَسْؤُولٌ بِالهَيْئَةِ العامَّةِ لِلرّقابَةِ المالِيَّةِ أَنّ الهَيْئَةِ أَوْشَكَت عَلَى الاِنْتِهاءِ مِن إِعْدادِ مَشْرُوعِ قانُون صَنادِيق التَّأْمِين الخاصَّةِ وَالمَعاشاتِ الاِخْتِيارِيَّةِ.

Finally, it is worthy to mention that that tagger has the ability to run with an average speed of 2000 words per minute, i.e. the tagger could assign the diacritization, tokenization and morphological analysis with a very high competitive speed. This speed can be enhanced by loading the resources once before running the tagging process for several times for several inputs.

7 CONCLUSIONS AND FUTURE WORK

The objective of this work has been the use of machine learning to design an Arabic automatic tagger that acts as a diacritizer, tokenizer and morphological analyzer. This objective has been reached through a supervised tagging technique, i.e., a one that uses an annotated corpus (whose correct Buckwalter morphological analysis is known) for classifiers training.
The results are promising and competitive. The tagger can assign the complete Buckwalter POS, English gloss, vocalization (diacritization), lemma ID and tokenization with accuracies of 85.7%, 92.3%, 87.7%, 95.2% and 99.6% respectively. The overall accuracy of assigning a complete Buckwalter analysis is 84.3%.

Enhancement and extension of the proposed system will be the subjects of future research work. For instance, the main problem in building corpora is the annotation process. If the annotations can be made automatically then it would save much time and efforts. This is doable by building a corpus whose annotations are made automatically using the presented tagger. Also, the annotations would be much rich and useful since the presented tagger can accomplish diacritization, tokenization and morphological analyses for the corpus text.

After syntactically analyzing the text, the next step is to analyze the text semantically and understand what does an input text means. A semantic analysis can be accomplished based on the syntactical analysis of the presented tagger. Semantic applications have great interest due to their importance in the real life practice, such as understanding the written/spoken text, monitoring and tracking systems, language learning programs and other several vital applications.

REFERENCES

[1] Imad A. Al-Sughaiyer and Ibrahim A. Al-Kharashi, "Arabic morphological analysis techniques. A comprehensive survey", Journal of the American Society for Information Science and Technology, Pages 189–213, 2004.

[2] Tim Buckwalter, "Buckwalter Arabic Morphological Analyzer", Linguistic Data Consortium, University of Pennsylvania, LDC (The Linguistic Data Consortium), Pages 1-5, 2002.

[3] Emad Mohamed and Sandra Kübler, "Is Arabic part of speech tagging feasible without word segmentation?", The 2010 Annual Conference of NAACL (North American Chapter of the Association for Computational Linguistics), Los Angeles, California, Pages 705-708, 2010.

[4] Kareem Darwish, "Building a shallow Arabic Morphological Analyser in One Day", 41st Meeting of ACL (Association for Computational Linguistics) Workshop on Computational Approaches to Semitic Languages, Philadelpia, PA, Pages 47-54, 2002.

[5] Young-Suk Lee, Kishore Papineni, Salim Roukos, Ossama Emam, and Hany Hassan, "Language model based Arabic word segmentation", 41st Meeting of ACL (Association for Computational Linguistics), Sapporo, Japan, Pages 399–406, 2003.

[6] Taku Kudo and Yuji Matsumato, "Use of Support Vector Learning for Chunk Identification", Proc. of the 4th Conf. on Very Large Corpora, Pages 142-144, 2000.

[7] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky, "Automatic tagging of Arabic Text: From Raw Text to Base Phrase Chunks", I5th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference, Boston, MA, Pages 149-152, 2004.

[8] Nizar Habash and Owen Rambow, Arabic Tokenization, "Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop", 43rd Meeting of ACL (Association for Computational Linguistics) Workshop on Computational Approaches to Semitic Languages, Pages 573-580, 2005.

[9] Shereen Khoja, "APT: Arabic Part-of-speech Tagger", Proc. of the Student Workshop at NAACL (North American Chapter of the Association for Computational Linguistics), Pages 20-25, 2001.

[10] Mohamed Maamouri, Ann Bies, and Tim Buckwalter, "The Penn Arabic Treebank: Building a Largescale Annotated Arabic Corpus", NEMLAR (Network for Euro-Mediterranean Language Resources) Conference on Arabic Language Resources and Tools, Cairo, Egypt, Pages 263–270, 2004.

[11] Taku Kudo and Yuji Matsumoto, "Fast Methods for Kernel-Based Text Analysis", 41st Meeting of ACL (Association for Computational Linguistics), Sapporo, Japan, Pages 24-31, 2003.

[12] Ali Farghaly and Khaled Shaalan, "Arabic Natural Language Processing: Challenges and Solutions", ACM Transactions on Asian Language Information Processing (TALIP), Volume 8, Issue 4, 2009.

المعايير اللغوية والفنية لتقييم
برامج التعرف الضوئي على الحروف العربية

عمرو جمعة عبد الرسول
كلية دار العلوم - جامعة القاهرة
amrdaramy@hotmail.com
amr1979go@yahoo.com

ملخص: تعد عملية رقمنة مصادر المعلومات العربية أمرا بالغ الأهمية، حيث يتم من خلالها تحويل نصوص مصادر المعلومات الورقية إلي شكل إلكتروني قابل لعليات المعالجة الآلية، وذلك من خلال المسح الضوئي للنصوص، ومن ثم الوصول إلى نسخة إلكترونية مطابقة تماماً للنص الأصلي الموجود في صورة ملفات jpg. أو pdf. على سبيل المثال.

ومع ازدياد الحاجة إلى رقمنة مثل هذه الملفات، من خلال التقنيات المتخصصة في التعرف الضوئي على الحروف "OCR" Optical Character Recognition زادت الحاجة إلى تحديد معايير لغوية لتقييم هذه البرامج.

والبحث الذي بين أيدينا يرسم خطوطا عريضة للمعايير اللغوية - على وجه الخصوص – لتقييم مثل هذه البرامج، وتتباين هذه المعايير ما بين الفنية واللغوية؛ فالفنية منها تخص شكل الحرف العربي وأنواع خطوطه المختلفة سواء باليد أو بالآلة الكاتبة، أما المعايير اللغوية فتخص الجانب اللغوي للحروف العربية من حيث اتصالها وانفصالها والبدء بها أو الانتهاء بها، هذا بالإضافة إلى ضبط الحروف العربية بالشكل في المستوى الكلاسيكي منها، وهو الأمر الذي يزيد من تعقيد عملية التعرف على الحروف العربية، ومن المعايير اللغوية كذلك مدى استخدام القواعد اللغوية في التعرف على الحروف العربية، أو استخدام آلية لغوية لتصحيح الأخطاء اللغوية في التعرف.

الكلمات الجوهرية

الخطوط العربية - الخط العربي - القارئ الآلي – التعرف الآلي على الحروف (المحارف) – التعرف الضوئي على الحروف – رقمنة مصادر المعلومات - Optical Character Recognition- "OCR"

وسنعرض في هذا البحث للنقاط التالية:

مشكلات التعرف على الحروف العربية، وتكمن في:

1- **الكتابة المتصلة للحروف العربية**، حيث تتميز العربية بكتابة حروفها متصلة غير منفصلة مثل الإنجليزية مثلا ، ومن ثم يصعب التعرف عليها نتيجة لتداخلها.

2- **نقط الحروف أو الإعجام**، تشترك نصف حروف العربية في نقط الحروف أو إعجامها، فالنقط مهم جدا في الأبجدية العربية، وذلك أنه يفرق بين الحروف، والعرب قديما كانوا يطلقون عليه الإعجام أي إزالة الغموض ويأتي في مقابله الإهمال، فالحروف إما مهملة أو معجمة. ولكن هذا النقط أو الإعجام يمثل صعوبة كبيرة عند عملية التعرف الآلي على الحروف العربية لتشابه الكثير من الحروف العربية في النقط. **فالحروف العربية (ج،ح،خ)**، لا يكاد يفرق بينها إلا التنقيط، فحروف كالجيم والحاء والخاء تكتب هكذا (ح) مع اختلاف وضع النقطة على الحرف أو عدم وضعها، وهو ما يضاف إلى صعوبات التعرف الآلي على الحروف.

والجدول التالي يرصد بعض صور هذه الحروف

احتمال 1		ب	ت	جـ	د	ر	شـ	صـ	طـ	عـ	فـ	ـه	ي
احتمال 2		يـ	ن	حـ	ذ	ز	ثـ	ضـ	ظـ	غـ	قـ	ـة	ى
احتمال 3				خـ			تـ			فـ	عـ		
احتمال 4										قـ	غـ		

جدول يوضح بعض صور الحروف المعجمة والمهملة في اللغة العربية

3- **الضبط بالشكل**، تتميز العربية بالتعبير عن الأصوات بحركات التشكيل، وتأتي هذه الحركات مقابلة للصوائت في اللغات الأخرى (e-o-i)، فحركات الكسرة والضمة والفتحة والسكون (ِ ُ َ ْ) كلها علامات على الصوائت وأدلة على وجودها. وهكذا يعتور الكلمة العربية الكثير من الزوائد الشكلية ابتداء من نقط الحروف وانتهاء بحركات الضبط بالشكل.

4- **الخطوط الفنية الزخرفية (كوفي – أندلسي – فارسي - ...)**، تتميز العربية بإمكانية كتابة حروفها بعدد من الحروف الزخرفية، والتي تضيف إلى الصعوبات السابقة المتعلقة بخصائص الحروف العربية صعوبة أخرى جديدة.

5- **تغير أشكال الحروف العربية بتغير مواقعها وتغير أشكالها بتغير نوع الخط المكتوبة به** فحرف (س) على سبيل المثال يكتب في أول الكلام (سـ) وفي آخر الكلمة (س) وفي وسط الكلمة (ـسـ)، وكذا يختلف شكله بخط النسخ عنه في حال خط الرقعة.

وثمة مستويان للنصوص المعالجة:

<u>أولا : النصوص غير المضبوطة بالشكل</u> (نصوص بسيطة)

<u>ثانيا: النصوص المضبوطة بالشكل</u> (نصوص معقدة)

<u>المعايير اللغوية والفنية لتقييم برامج التعرف الآلي على الحروف العربية:</u>

<u>المعيار الأول:</u> التعرف على جميع حروف الخطوط العربية المستخدمة في الكتابة العربية

<u>وإننا نقترح المصادر التالية لتكون مصادر نموذجية لعملية التقييم:</u>

<u>مصادر تجميع النصوص (من حروف وأرقام وعلامات تشكيل وعلامات ترقيم) المستخدمة لتقييم برامج التعرف الآلي على الحروف العربية:</u>

(1) مجموعة من خطوط اليد لمجموعة متفاوتة في السن وفي التعليم.

(2) مجموعة مختلفة الخطوط من الصحف والجرائد في شتى أنحاء الوطن العربي.

(3) مستندات مكتوبة على الآلة الكاتبة.

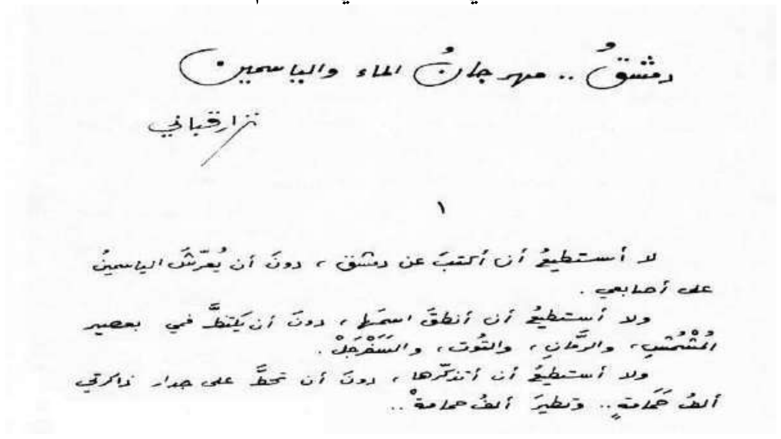
(4) نصوص لخطاطين بشتى أنواع الخط العربي غير الزخرفية.

(5) نماذج من خطوط الأوفيس غير الزخرفية.

(6) تنسيقات مختلفة لصور الخطوط السابقة جميعها. فاحتواء النص على تنسيقات غاية في التعقيد مثل (وجود عدة أعمدة رأسية، ووجود إيضاحات أو هوامش أو

حواشٍ في أماكن غير منتظمة، إلي غير ذلك)، من شأنه أن يكون له تأثير سلبي على جودة التعرف الضوئي على الحروف.
(7) نصوص متعددة اللغات أو متعددة الرموز أو الحروف (حروف وأرقام متداخلة، أو تداخل العديد من اللغات في النصوص التي تحتوي أكثر من لغة).

وسنحاول في الصفحات القليلة القادمة التمثيل لكل من النماذج السابقة:
(1) مجموعة من خطوط اليد لمجموعة متفاوتة في السن وفي التعليم.

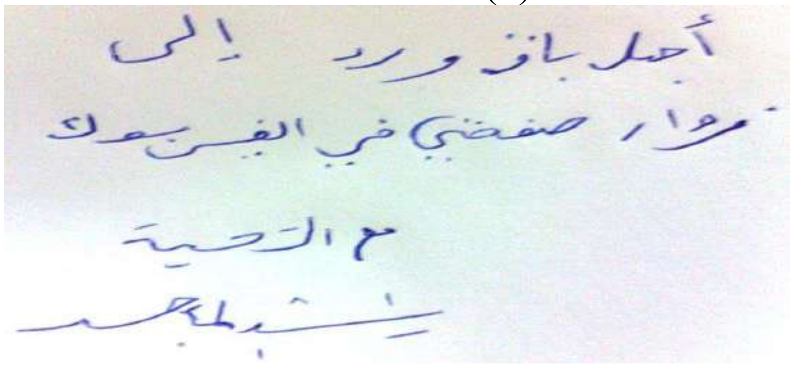

دمشقُ .. مهرجانُ الماء والياسمين

نزار قباني

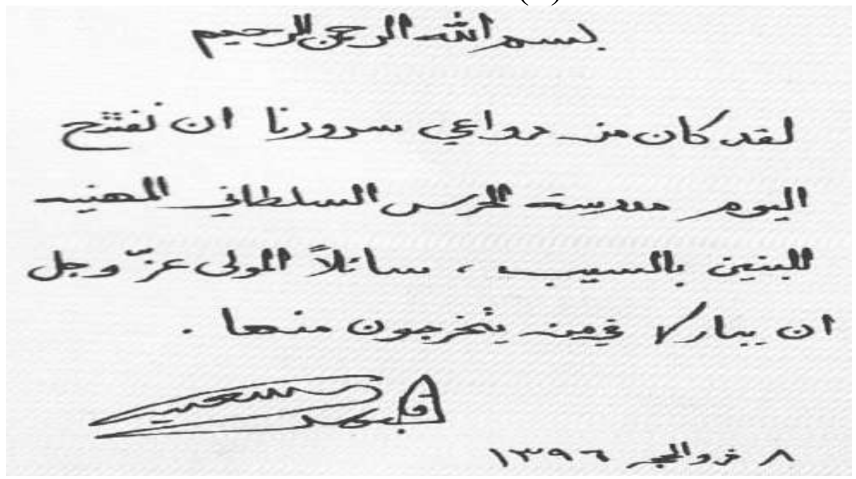
١

لا أستطيعُ أن أكتبَ عن دمشق، دونَ أن يُعرِّشَ الياسمينُ على أصابعي.
ولا أستطيعُ أن أنطقَ اسمَها، دونَ أن يَكتظَّ فمي بعصيرِ المشمشِ، والرمانِ، والتوتِ، والسفرجلْ.
ولا أستطيعُ أن أتذكّرها، دونَ أن تحطَّ على جدارِ ذاكرتي ألفُ حمامةٍ.. وتطيرَ ألفُ حمامةْ..

شكل (1) يمثل نموذجا لخط اليد

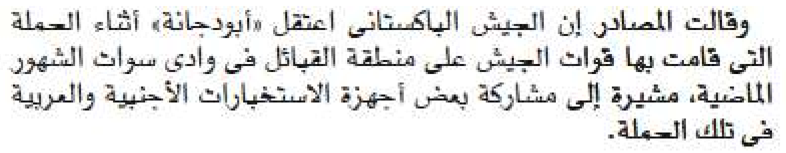

أجمل باقة ورد إلى
زوار صفحتي في الفيسبوك
مع التحية
[illegible]

شكل (2) يمثل نموذجا لخط اليد


بسم الله الرحمن الرحيم
لقد كان من دواعي سرورنا ان نفتتح
اليوم مدرسة الحرس السلطاني المهنية
للبنين بالسيب، سائلاً المولى عزّ وجل
ان يبارك فيمن يتخرجون منها.
[illegible]
٨ ذو الحجة ١٣٩٦

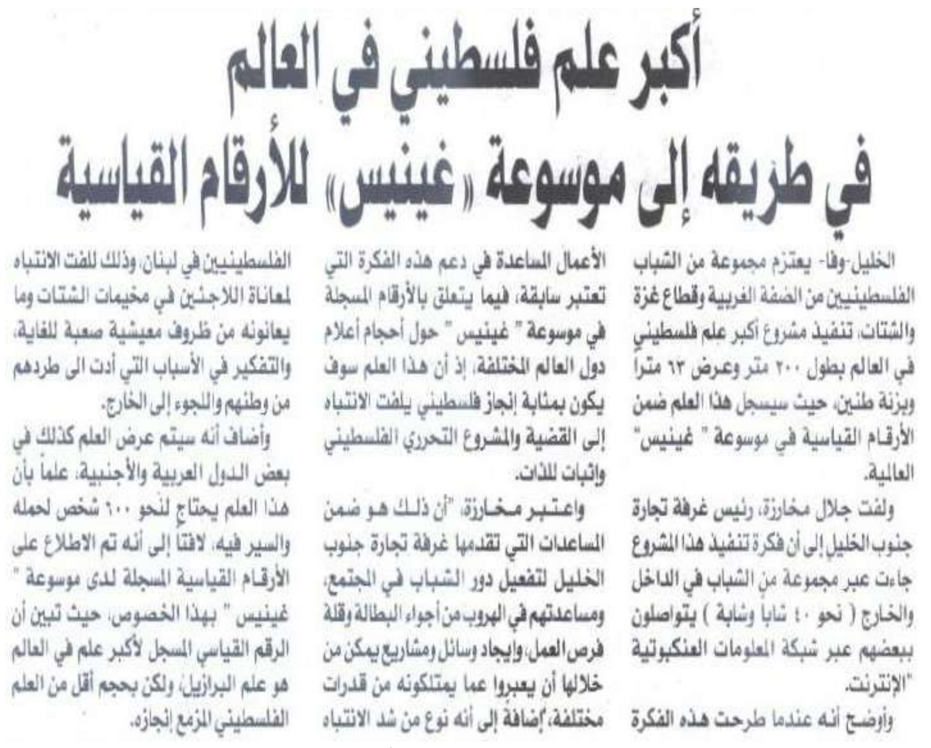
شكل (3) يمثل نموذجا لخط اليد

(2) مجموعة مختلفة الخطوط من الصحف والجرائد في شتى أنحاء الوطن العربي.



وقالت المصادر إن الجيش الباكستانى اعتقل «أبودجانة» أثناء الحملة التى قامت بها قوات الجيش على منطقة القبائل فى وادى سوات الشهور الماضية، مشيرة إلى مشاركة بعض أجهزة الاستخبارات الأجنبية والعربية فى تلك الحملة.

شكل (4) يمثل نموذجا لخط إحدى الصحف



أكبر علم فلسطيني في العالم في طريقه إلى موسوعة «غينيس» للأرقام القياسية

الخليل-وفا- يعتزم مجموعة من الشباب الفلسطينيين من الضفة الغربية وقطاع غزة والشتات، تنفيذ مشروع أكبر علم فلسطيني في العالم بطول ٢٠٠ متر وعرض ٦٣ متراً وبزنة طنين، حيث سيسجل هذا العلم ضمن الأرقام القياسية في موسوعة " غينيس" العالمية.

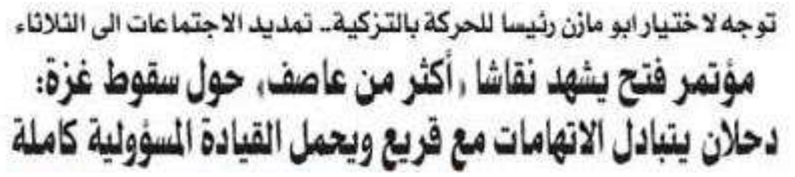
ولفت جلال مخارزة، رئيس غرفة تجارة جنوب الخليل إلى أن فكرة تنفيذ هذا المشروع جاءت عبر مجموعة من الشباب في الداخل والخارج (نحو ٤٠ شاباً وشابة) يتواصلون ببعضهم عبر شبكة المعلومات العنكبوتية "الإنترنت.

وأوضح أنه عندما طرحت هذه الفكرة الأعمال الساعدة في دعم هذه الفكرة التي تعتبر سابقة، فيما يتعلق بالأرقام المسجلة في موسوعة " غينيس " حول أحجام أعلام دول العالم المختلفة، إذ أن هذا العلم سوف يكون بمثابة إنجاز فلسطيني يلفت الانتباه إلى القضية والمشروع التحرري الفلسطيني واثبات للذات.

واعتبر مخارزة، "أن ذلك هو ضمن الساعدات التي تقدمها غرفة تجارة جنوب الخليل لتفعيل دور الشباب في المجتمع، ومساعدتهم في الهروب من أجواء البطالة وقلة فرص العمل، وإيجاد وسائل ومشاريع يمكن من خلالها أن يعبروا عما يمتلكونه من قدرات مختلفة، إضافة إلى أنه نوع من شد الانتباه الفلسطينيين في لبنان، وذلك للفت الانتباه لمعاناة اللاجئين في مخيمات الشتات وما يعانونه من ظروف معيشية صعبة للغاية، والتفكير في الأسباب التي أدت الى طردهم من وطنهم واللجوء إلى الخارج.

وأضاف أنه سيتم عرض العلم كذلك في بعض الدول العربية والأجنبية، علماً بأن هذا العلم يحتاج لنحو ٦٠٠ شخص لحمله والسير فيه، لافتا إلى أنه تم الاطلاع على الأرقام القياسية المسجلة لدى موسوعة " غينيس " بهذا الخصوص، حيث تبين أن الرقم القياسي المسجل لأكبر علم في العالم هو علم البرازيل، ولكن بحجم أقل من العلم الفلسطيني المزمع إنجازه.

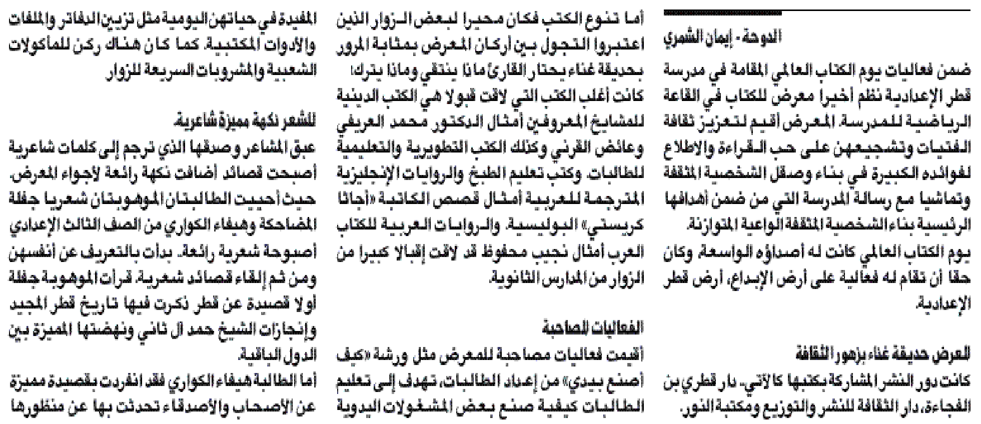
شكل (5) يمثل نموذجا لخط إحدى الصحف



توجه لاختيار ابو مازن رئيسا للحركة بالتزكية- تمديد الاجتماعات الى الثلاثاء

مؤتمر فتح يشهد نقاشا ،أكثر من عاصف، حول سقوط غزة:
دحلان يتبادل الاتهامات مع قريع ويحمل القيادة المسؤولية كاملة

شكل (5) يمثل نموذجا لخط إحدى الصحف



الدوحة - إيمان الشمري

ضمن فعاليات يوم الكتاب العالمي المقامة في مدرسة قطر الإعدادية نظم أخيرا معرض للكتاب في القاعة الرياضية للمدرسة. المعرض أقيم لتعزيز ثقافة الفتيات وتشجيعهن على حب القراءة والإطلاع لفوائده الكبيرة في بناء وصقل الشخصية المثقفة وتماشيا مع رسالة المدرسة التي من ضمن أهدافها الرئيسية بناء الشخصية المثقفة الواعية المتوازنة.
يوم الكتاب العالمي كانت له أصداؤه الواسعة، وكان حقا أن تقام له فعالية على أرض الإبداع، أرض قطر الإعدادية.

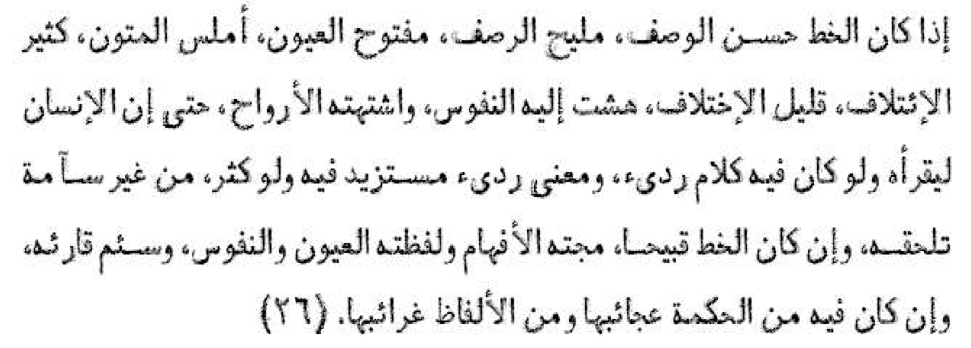
المعرض حديقة غناء بزهور الثقافة
كانت دور النشر المشاركة بكتبها كالآتي. دار قطري بن الفجاءة، دار الثقافة للنشر والتوزيع ومكتبة النور.
أما تنوع الكتب فكان محيرا لبعض الزوار الذين اعتبروا التجول بين أركان المعرض بمثابة المرور بحديقة غناء يحتار القارئ ماذا ينتقي وماذا يترك!
كانت أغلب الكتب التي لاقت قبولا هي الكتب الدينية للمشايخ المعروفين أمثال الدكتور محمد العريفي وعائض القرني وكذلك الكتب التطويرية والتعليمية للطالبات. وكتب تعليم الطبخ والروايات الإنجليزية المترجمة للعربية أمثال قصص الكاتبة «أجاثا كريستي» البوليسية. والروايات العربية للكتاب العرب أمثال نجيب محفوظ قد لاقت إقبالا كبيرا من الزوار من المدارس الثانوية.

الفعاليات المصاحبة
أقيمت فعاليات مصاحبة للمعرض مثل ورشة «كيف أصنع بيدي» من إعداد الطالبات، تهدف إلى تعليم الطالبات كيفية صنع بعض المشغولات اليدوية المفيدة في حياتهن اليومية مثل تزيين الدفاتر والملفات والأدوات المكتبية. كما كان هناك ركن للماكولات الشعبية والمشروبات السريعة للزوار

للشعر نكهة مميزة شاعرية
عبق المشاعر وصدقها الذي ترجم إلى كلمات شاعرية أصبحت قصائد أضافت نكهة رائعة لأجواء المعرض. حيث أحييت الطالبتان الموهوبتان شعريا جفلة المضاحكة وهيفاء الكواري من الصف الثالث الإعدادي أصبوحة شعرية رائعة. بدأت بالتعريف عن أنفسهن ومن ثم إلقاء قصائد شعرية. قرأت الموهوبة جفلة أولا قصيدة عن قطر ذكرت فيها تاريخ قطر المجيد وإنجازات الشيخ حمد آل ثاني ونهضتها المميزة بين الدول الباقية.
أما الطالبة هيفاء الكواري فقد انفردت بقصيدة مميزة عن الأصحاب والأصدقاء تحدثت بها عن منظورها

شكل (6) يمثل نموذجا لخط إحدى الصحف

(3) مستندات مكتوبة على الآلة الكاتبة.

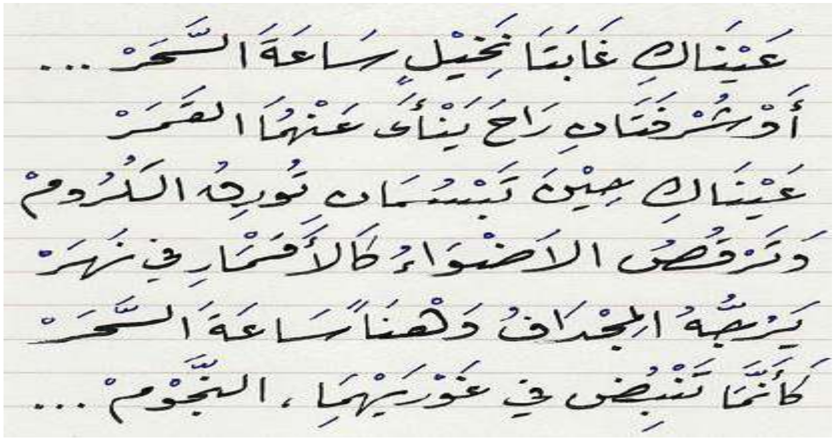


إذا كان الخط حسن الوصف، مليح الرصف، مفتوح العيون، أملس المتون، كثير الإئتلاف، قليل الإختلاف، هشت إليه النفوس، واشتهته الأرواح، حتى إن الإنسان ليقرأه ولو كان فيه كلام رديء، ومعنى رديء مستزيد فيه ولو كثر، من غير سآمة تلحقه، وإن كان الخط قبيحا، مجته الأفهام ولفظته العيون والنفوس، وسئم قارئه، وإن كان فيه من الحكمة عجائبها ومن الألفاظ غرائبها. (٢٦)

شكل (7) يمثل نموذجا لخط الآلة الكاتبة

(4) نصوص لخطاطين بشتى أنواع الخط العربي غير الزخرفية.

الخط الرقعي



عَيْنَاكِ غَابَتَا نَخِيلٍ سَاعَةَ السَّحَرْ ...
أَوْ شُرْفَتَانِ رَاحَ يَنْأَى عَنْهُمَا الْقَمَرْ
عَيْنَاكِ حِينَ تَبْسُمَانِ تُورِقُ الْكُرُومْ
وَتَرْقُصُ الأَضْوَاءُ كَالأَقْمَارِ فِي نَهَرْ
يَرُجُّهُ الْمِجْدَافُ وَهْنًا سَاعَةَ السَّحَرْ
كَأَنَّمَا تَنْبُضُ فِي غَوْرَيْهِمَا، النُّجُومْ ...

شكل (8) يمثل نموذجا لخط الرقعة مضبوطا بالشكل

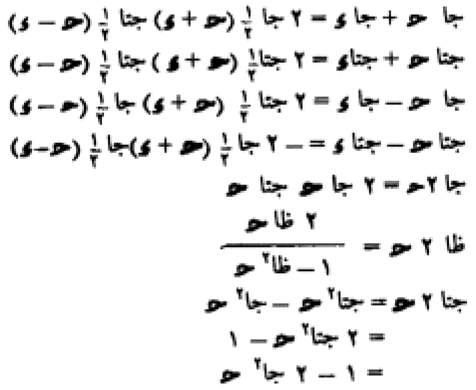
(5) نماذج من خطوط الأوفيس غير الزخرفية.

جدول (1) يوضح خطوط الأوفيس الزخرفية منها وغير الزخرفية

Office fonts			
Akhbar MT	بسم الله الرحمن الرحيم	Microsoft Uighur	بسم الله الرحمن الرحيم
Andalus	بسم الله الرحمن الرحيم	Monotype Koufi	بسم الله الرحمن الرحيم
Arabic Typesetting	بسم الله الرحمن الرحيم	Mudir MT	بسم الله الرحمن الرحيم
Arial	بسم الله الرحمن الرحيم	Old Antic Bold	بسم الله الرحمن الرحيم
Bold Italic Art	بسم الله الرحمن الرحيم	Old Antic Decorative	بسم الله الرحمن الرحيم
DecoType Naskh	بسم الله الرحمن الرحيم	Old Antic Outline	بسم الله الرحمن الرحيم
DecoType Naskh Extensions	بسم الله الرحمن الرحيم	Old Antic Outline Shaded	بسم الله الرحمن الرحيم
DecoType Naskh Special	بسم الله الرحمن الرحيم	PT Bold Arch	بسم الله الرحمن الرحيم
DecoType Naskh Swashes	بسم الله الرحمن الرحيم	PT Bold Broken	بسم الله الرحمن الرحيم
DecoType Naskh Variants	بسم الله الرحمن الرحيم	PT Bold Dusky	بسم الله الرحمن الرحيم
DecoType Thuluth	بسم الله الرحمن الرحيم	PT Bold Heading	بسم الله الرحمن الرحيم
Diwani Bent	بسم الله الرحمن الرحيم	PT Bold Mirror	بسم الله الرحمن الرحيم
Diwani Letter	بسم الله الرحمن الرحيم	PT Bold Stars	بسم الله الرحمن الرحيم
Diwani Outline Shaded	بسم الله الرحمن الرحيم	PT Simple Bold Ruled	بسم الله الرحمن الرحيم
Diwani Simple Outline	بسم الله الرحمن الرحيم	Simple Bold Jut Out	بسم الله الرحمن الرحيم
Diwani Simple Outline 2	بسم الله الرحمن الرحيم	Simple Indust Outline	بسم الله الرحمن الرحيم
Diwani Simple Striped	بسم الله الرحمن الرحيم	Simple Indust Shaded	بسم الله الرحمن الرحيم
Farsi Simple Bold	بسم الله الرحمن الرحيم	Simple Outline Pat	بسم الله الرحمن الرحيم
Farsi Simple Outline	بسم الله الرحمن الرحيم	Simplified Arabic	بسم الله الرحمن الرحيم
Italic Outline Art	بسم الله الرحمن الرحيم	Simplified Arabic Fixed	بسم الله الرحمن الرحيم
Kufi Extended Outline	بسم الله الرحمن الرحيم	Tahoma	بسم الله الرحمن الرحيم
Kufi Outline Shaded	بسم الله الرحمن الرحيم	Times New Roman	بسم الله الرحمن الرحيم
Led Italic Font	بسم الله الرحمن الرحيم	Traditional Arabic	بسم الله الرحمن الرحيم

وقد قمنا بتظليل الخطوط الزخرفية في قائمة خطوط الأوفيس؛ لتجنب معالجتها –على الأقل في المرحلة الأولى من معايير البرنامج-

(6) تنسيقات مختلفة لصور الخطوط السابقة جميعها. فاحتواء النص على تنسيقات غاية في التعقيد مثل (وجود عدة أعمدة رأسية، ووجود إيضاحات أو هوامش أو حواشٍ في أماكن غير منتظمة، إلي غير ذلك)، من شأنه أن يكون له تأثير سلبي على جودة التعرف الضوئي على الحروف.



جا حـ + جا ى = ٢ جا ١/٢ (حـ + ى) جتا ١/٢ (حـ − ى)

جتا حـ + جتاى = ٢ جتا١/٢ (حـ + ى) جتا ١/٢ (حـ − ى)

جا حـ − جا ى = ٢ جتا ١/٢ (حـ + ى) جا ١/٢ (حـ − ى)

جتا حـ − جتا ى = − ٢ جا ١/٢ (حـ + ى)جا ١/٢ (حـ−ى)

جا ٢حـ = ٢ جا حـ جتا حـ

ظا ٢ حـ = ٢ ظا حـ / (١ − ظا٢ حـ)

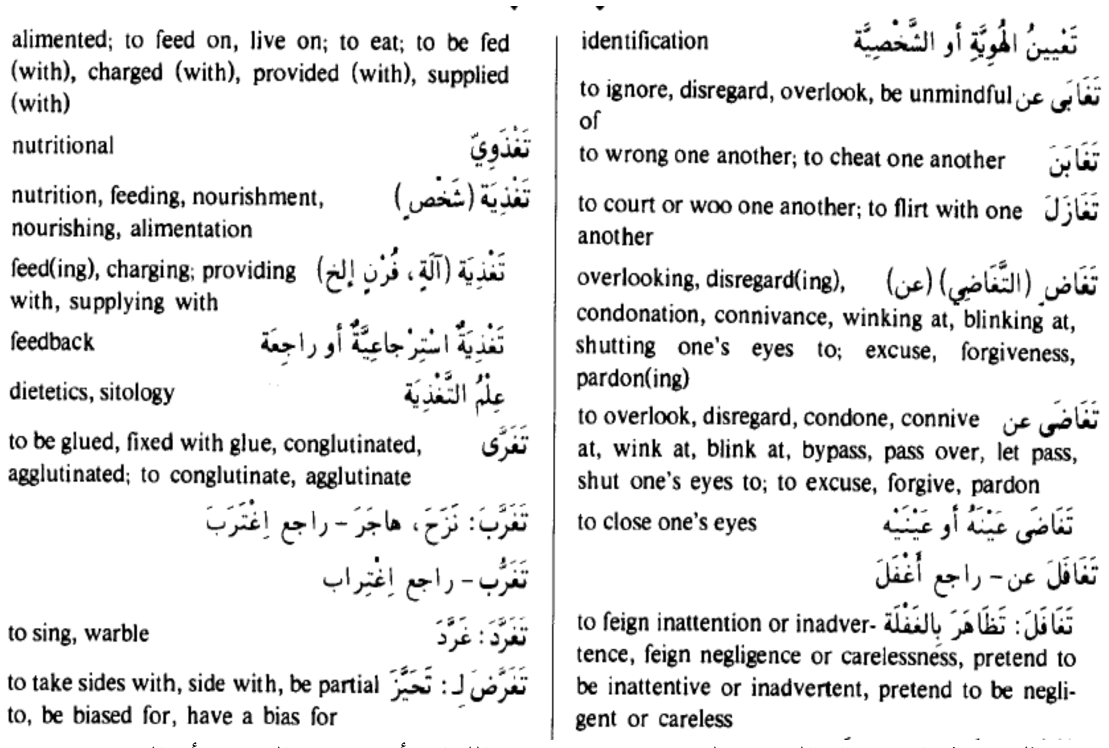
جتا ٢ حـ = جتا٢ حـ − جا٢ حـ

= ٢ جتا٢ حـ − ١

= ١ − ٢ جا٢ حـ

شكل (9) يمثل نموذجا لاحتواء النص على تنسيقات غاية في التعقيد

(7) نصوص متعددة اللغات أو متعددة الرموز أو الحروف (حروف وأرقام متداخلة ، أو تداخل العديد من اللغات في النصوص التي تحتوي أكثر من لغة).



تَعْيينُ الْهُوِيَّةِ أو الشَّخْصِيَّة	identification
تَغَابَى عن	to ignore, disregard, overlook, be unmindful of
تَغَابَنَ	to wrong one another; to cheat one another
تَغَازَلَ	to court or woo one another; to flirt with one another
تَغَاضٍ (التَّغَاضِي) (عن)	overlooking, disregard(ing), condonation, connivance, winking at, blinking at, shutting one's eyes to; excuse, forgiveness, pardon(ing)
تَغَاضَى عن	to overlook, disregard, condone, connive at, wink at, blink at, bypass, pass over, let pass, shut one's eyes to; to excuse, forgive, pardon
تَغَاضَى عَيْنَهُ أو عَيْنَيْه	to close one's eyes
تَغَافَلَ عن – راجع أَغْفَلَ	
تَغَافَلَ: تَظَاهَرَ بِالغَفْلَة	to feign inattention or inadvertence, feign negligence or carelessness, pretend to be inattentive or inadvertent, pretend to be negligent or careless

	alimented; to feed on, live on; to eat; to be fed (with), charged (with), provided (with), supplied (with)
تَغَذَوِيّ	nutritional
تَغْذِيَة (شَخْصٍ)	nutrition, feeding, nourishment, nourishing, alimentation
تَغْذِيَة (آلَةٍ، فُرْنٍ إلخ)	feed(ing), charging; providing with, supplying with
تَغْذِيَةٌ اسْتِرْجاعِيَّةٌ أو راجِعَة	feedback
عِلْمُ التَّغْذِيَة	dietetics, sitology
تَغَرَّى	to be glued, fixed with glue, conglutinated, agglutinated; to conglutinate, agglutinate
تَغَرَّبَ: نَزَحَ، هاجَرَ – راجع اِغْتَرَبَ	
تَغَرُّب – راجع اِغْتِراب	
تَغَرَّدَ: غَرَّدَ	to sing, warble
تَغَرَّضَ لِـ: تَحَيَّزَ	to take sides with, side with, be partial to, be biased for, have a bias for

شكل (10) يمثل نموذجا لاحتواء النص على نصوص متعددة اللغات أو متعددة الرموز أو الحروف

<u>المعيار الثاني:</u> التعرف على علامات الضبط بالشكل (علامات التشكيل) والتفرقة بينها وبين نقط الإعجام

تتميز اللغة العربية بالتعبير عن الأصوات (vowels) بحركات الضبط، وتأتي هذه الحركات مقابلة للصوائت في اللغات الأخرى (e-o-i-u-a)، وهكذا يتعاور على الكلمة العربية الكثير من الزوائد الشكلية من تنقيط الحروف وحركات الضبط بالشكل.

جدول (2) يوضح علامات الضبط بالشكل في العربية

◌َ	◌ً	◌ُ	◌ٌ	◌ِ	◌ٍ	◌ْ	◌◌َّ	◌◌ًّ	◌◌ُّ
فتحة	**تنوين بالفتحة**	**ضمة**	**تنوين بالضمة**	**كسرة**	**تنوين بالكسرة**	**سكون**	**شدة وفتحة**	**شدة وتنوين بالفتحة**	**شدة وضمة**
◌◌ٌّ	◌◌ِّ	◌◌ٍّ							
شدة وتنوين بالضمة	**شدة وكسرة**	**شدة وتنوين بالكسرة**							

والحق أنها اللغة العربية المعاصرة لا تستخدم علامات الضبط بالشكل في النصوص المعاصرة، ومن ثم يمكن مراعاة هذا المعيار في النصوص الكلاسيكية وحسب، وتجاهله في النصوص المعاصرة.

<u>المعيار الثالث:</u> التعرف على الأرقام العربية بصورتيها الهندية (١-٢-٣) والعربية (1-2-3) .

<u>المعيار الرابع:</u> التعرف على علامات الترقيم (، - . - ؛ - ؟ - !). ومن المهم هنا ملاحظة أن هذه العلامات قد تتصل بالكلمة تارة وقد لا تتصل بها تارة أخرى، ومن ثم ينبغي معالجة هذه العلامات في كلتا صورتيها.

<u>المعيار الخامس:</u> التعرف على الرموز والاختصارات المختلفة (@ - # - $ - % - [...])

<u>المعيار السادس:</u> التعرف على الصور والأشكال الرسومية، ونقلها مرة أخرى إلى الملف الرقمي في مكانها من الملف الأصلي.

<u>المعيار السابع:</u> مدى استخدام المعارف اللغوية في عملية التعرف الآلي على الحروف (من خلال مراعاة خصائص الحروف من حيث الشكل والترتيب والضبط بالشكل)

فللحروف العربية خصائص من حيث الشكل والموضع من الكلمة، فهناك حروف تأتي في أول الكلمة فقط، وهناك حروف تأتي في أوسطها ومنتصفها فقط ، وهناك حروف متطرفة تأتي في أواخر الكلمة فقط (الحروف الاستهلالية - الحروف المتوسطة - الحروف المتطرفة)، بل يتغير رسم الحرف باختلاف ترتيبه مع الحروف الأخرى بداية أو توسطا أو نهاية.

ومثال ذلك:

حرف الألف (الهمزة)

تختلف أشكال كتابة **الألف (الهمزة)** كما يلي - على سبيل الحصر- :

{أ – إ – ا – ء – آ – ؤ – ئ – ى}

فجميع هذه الصور تأتي في بداية الكلمة ماعدا

{ء – ؤ – ئ – ى}

وجميعها أيضا تأتي في وسط الكلمة ماعدا

{ى}

وجميعها كلها تأتي في نهاية الكلمة ماعدا

{إ}

وقد تجتمع أكثر من همزة في بداية الكلمة ولكن على النحو التالي فقط :

{أأ – أئ - أؤ - أ – آل}

ومن ذلك أيضا مراعاة قواعد الرسم الإملائي في العربية، ومنها:

- **توالي الأمثال في العربية**

تهرب العربية من توالي الأمثال على مستوى الحرفين أحيانا، وتهرب منه بشكل قاطع على مستوى ثلاثة أحرف.

ويمكن حصر الحروف التي لا تتكرر أبدا على مستوى حرفين، ومن ثم تخطئتها إذا جاءت مكررة، أما توالي 3 حروف فممنوع في العربية منعا قاطعا مثل(لللـ) في (لللیمون) وصوابها (للیمون)، ومن ثم يمكن حذف الحرف المكرر الثالث تلقائيا أو استبداله بما يشابهه.

ومن ذلك أيضا – عند التعرف على النصوص المضبوطة بالشكل- **مراعاة قواعد الضبط والتشكيل:**

أ- الفتحة والكسرة والضمة والسكون تأتي مع جميع مواضع الحروف من الكلمة (الحرف الأول والثاني والثالث، ... إلخ

ب- التنوين بالفتحة والكسرة والضمة يأتي مع نهايات الكلمة فقط (على الحرف الأخير) ولا يأتي التنوين مع بداية الكلمة (على الحرف الأول) أو في وسطها (وقد يأتي التنوين قبل الآخر في حال التنوين بالفتحة فقط مع إحدى صور كتابة التنوين بالفتحة)

ج- التنوين بالفتحة يزيد حرفا هو الألف على الكلمة غير المنتهية بتاء مربوطة أو ياء.

د- السكون لا يأتي على الحرف الأول في الكلمة في العربية والقاعدة اللغوية في ذلك مشهورة: لا يُبدأ بساكن .

هـ - لا يتوالى **ساكنان** في العربية (ـْ ـْ)

و- الشدة لا تأتي على الحرف الأول في الكلمة في العربية ذلك أن الشدة عبارة عن حرفين: حرف ساكن+ حرف متحرك

ز- ألف الوصل لا تضبط بالشكل

<u>المعيار الثامن:</u> وجود تقنية لغوية متقدمة تمكنها من التصحيح التلقائي لأخطاء القراءة بالاعتماد على مدقق إملائي يعتمد على محلل صرفي) أو من خلال (معجم لغوي)

<u>المعيار التاسع:</u> مدى الاستفادة من تصحيحات المستخدم وتدريب برامج التعرف الآلي على التعلم من أخطائه السابقة

مواقع الإنترنت [من مواقع (برامج التعرف الآلي على الحروف) على شبكة الإنترنت]

http://www.caere.com
http://www.irislink.com
http://www.sakhr.com
http://www.textbridge.com
http://www.novodynamics.com

http://www.ocr-systeme.de
http://www.olduvai.com

المراجع والمصادر

أولا: المراجع اللغوية

1. ابن يعيش، شرح المفصل، تحقيق د. إميل يعقوب، دار الكتب العلمية، ط.1، 1422هـ، 2001م .
2. أبو الوفاء نصر الهوريني، المطالع النصرية في الأصول الخطية للمطابع المصرية، دار أضواء السلف للنشر والتوزيع، ط (1)، 1426هـ/ 2005م.
3. الرضي الإستراباذي، شرح شافية ابن الحاجب، تحقيق د. إميل يعقوب، دار الكتب العلمية، ط. 1، 1419هـ.
4. رمضان عبد التواب، مشكلة الهمزة العربية، مكتبة الخانجي، القاهرة، ط. الأولى، 1417هـ1996 /م.
5. سليمان فياض، استخدامات الحروف العربية، دار المريخ ، 1998
6. عبد الجواد الطيب، دراسة في قواعد الإملاء، دار الأوزاعي، ط. ثانية، بيروت 1406هـ/1986م.
7. عبد السلام هارون، قواعد الإملاء، دار إيلاف الدولية، الكويت، ط. الأولى، 1425هـ/ 2004م.
8. عبد العليم إبراهيم، الإملاء والترقيم في الكتابة العربية، مكتبة غريب، القاهرة، 1975م.
9. محمد الحملاوي، شذا العرف في فن الصرف، طبعة مؤسسة الرسالة للطباعة والنشر والتوزيع، 2003.
10. محمود حاج حسين، تاريخ الكتابة العربية وتطورها وأصول الإملاء العربي، وزارة الثقافة، دمشق 2004م.
11. مصطفى التوني، الهمزة في اللغة العربية، دراسة لغوية، القاهرة 1990م.

ثانيا: الرسائل الجامعية

12. العربي محمد إبراهيم، منظومة الصرف العربي والمعالجة الحاسوبية، دراسة تقويمية – رسالة ماجستير بكلية دار العلوم رقم 1488.

ثالثا: الأبحاث

13. أحمد فرج أحمد، تقنيات التعرف الضوئي للحروف، معايير الاختيار، طريقة العمل، الإشكاليات، والآفاق المستقبلية، جامعة الإمام محمد بن سعود الإسلامية، كلية علوم الحاسب والمعلومات، قسم دراسات المعلومات.
14. محمد زكي خضر ، التقاء الحروف المتماثلة في القرآن الكريم، مجلة الفرقان، العدد 50، آذار 2006.

محمد زكي خضر، الحرف العربي والحوسبة، الموسم الثقافي لمجمع اللغة العربية، عمان – الأردن ، 2001.

Rich Semantic Graph Generation System Prototype

Mostafa Mahmoud Aref [*1], Ibrahim Fathy Moawad[**2], Soha Said Ibrahim[*3]

[*]*Computer Science Department, Faculty of Computer Science and Information Sciences, Ain Shams University*
Abbassia, Cairo, Egypt

[1]aref_99@yahoo.com

[3]sohaelshafey@yahoo.com

[**]*Information System Department, Faculty of Computer Science and Information Sciences, Ain Shams University*
Abbassia, Cairo, Egypt

[2]ibrahim_moawad@hotmail.com

***Abstract*— Information nowadays has become more and more accessible, so much as to give birth to an information overload issue. As it is impossible to read all the relevant content that helps one stay informed, a possible solution would be condensing data and obtaining the kernel of a text to explore, analyze, and discover knowledge from documents. In this paper, a system prototype is proposed to analyze text and represent valuable information in an ontology-based representation form. The main objective is to propose a system that transforms an input text to semantic graph representation called "Rich Semantic Graph" (RSG) in which verbs and nouns of a document are represented as nodes along with edges corresponding to semantic and linguistic relations between them. The rich semantic graph can be exploited in several applications like information retrieval, text summarization, and text mining applications. This work is a part of an ongoing research to create an abstractive summary for a single input document.**

1 INTRODUCTION

The explosive growth in the number of electronic documents produced daily increases the need for intelligent filtering approaches. It becomes impossible to manually search, sift and choose the needed information. Knowledge representation and discovery is the non-trivial extraction of implicit, unknown, and potentially useful information from document such that it is easily accessed and manipulated [1], [2]. Different Natural Language Processing (NLP) applications (e.g. text summarization, machine translation etc.) entail different internal representations that lead to different implementation techniques. Therefore, a common semantic representation is needed to unify application representation techniques. Ontology has evolved in computer science as a formal explicit specification of a shared conceptualization of vocabularies. It provides a shared and a common understanding of vocabulary that can be communicated between people and distributed systems. By defining shared and common vocabularies, ontology helps both people and machines to communicate and support the exchange of semantics but also syntax [3].

In this paper, a new ontology-based system prototype is proposed to generate unified semantic representation from the input document. The provided representation consists of document concepts as nodes along with edges corresponding to semantic and linguistic relationships between them. This semantic graph representation is called "Rich Semantic Graph" (RSG). It can be exploited in several applications like information retrieval, text summarization, and text mining applications. This work is part of a novel approach to create an abstractive summary for a single input document using the "Rich Semantic Graph" (RSG). The method summaries the input single document by creating a rich semantic graph for the original document, reducing the generated graph to smaller graph, then applying a natural language generation technique on the reduced graph to generate the final abstractive summary [4],[5]. In section 2, problem definition and related work are presented, while section 3 presents the RSG generation model. Section 4 describes the system prototype, and finally section 5 concludes the paper.

2 PROBLEM DEFINITION AND RELATED WORK

Several research activities are related to this work. One of the most recent research methods is extracting summary sentences based on the document semantic graph representation. This method generates semantic representation from the input document by a machine learning technique to extract full sentences suitable for creating summaries. It starts with deep syntactic analysis of the whole text, then for each sentence extracts logical form triples (subject-predicate-object). After that, it applies cross sentence pronoun resolution, co-reference resolution, and semantic normalization to refine the set of triples and merge them into a semantic graph. This procedure is applied to both document and corresponding summary extracts. Finally, linear support vector machine will be trained on the logical form triples to learn how to extract triples that belong to sentences in document summaries. The classifier is then used for automatic creation of document summaries of test documents [6] - [11].

On the other hand, another approach incorporates the object-orientation techniques in knowledge representation. This approach supports data abstraction and hence increases the modularity of natural language processing applications. It organizes knowledge into classes of objects (subclasses and super classes), which is an important issue in knowledge representation to avoid redundant declarations or specifications. This approach assumes the presence of an object-oriented lexicon where language words represent objects, and their semantic characteristics are expressed in terms of attributes and behaviors. For a given sentence, set of words, the model instantiates the word objects using some lexicon program. These word objects search for their suitable partners to come up with a semantically consistent structure that reflects the meaning of the sentence. At the end, the model gives an output of some weighted representations of the input sentence. These weights reflect the most likely meanings of the sentence [12], [13].

Finally, an Interlingua is proposed to represent and exchange information called "Universal Networking Language (UNL)". UNL is equipped with lexical, grammatical, and semantic components as any natural language. Therefore, it can describe the entire world of natural language. UNL system consists of Universal Words (UW), relations, attributes, and the UNL knowledge base (UNLKB). In the UNL approach, information conveyed by natural language is represented by a hyper graph. This hyper graph is composed of a set of hyper nodes (UWs) and directed binary labeled links (relations) between these nodes. The process of representing natural language sentences in UNL graphs is called "*the enconverting process"*, while the process of generating natural language sentences out of UNL graphs is called *"the deconverting process"* [14], [15], [21] - [23].

Nevertheless, semantic graph representation in its traditional form is incomplete due to the limitation of explicit operational or procedural knowledge, so it needs to assign more structure to nodes as well as to links [3]. On the other hand, object-oriented (OO) modeling reflects the structure of data and the software's object behavior not the world concepts and its structure. Therefore, only traditional object-oriented technique is not enough for a good knowledge representation. It is difficult to build large coherent and complete knowledge representation [16], [17]. Finally, the granularity is an important issue in knowledge representation. It concerns with the level of knowledge details that should be represented and what are the primitives (fundamental concepts) needed to be constructed [3]. UWs primitives in UNL consider most natural language elements: nouns, verbs, adjectives, and adverbs. However, the UNL hyper graph generated to represent a short document will be a very huge due to the low granularity of UNL system concepts.

In this paper, a system prototype is presented to transform an input document to a Rich Semantic Graph (RSG) which is based upon ontology. The ontology primitives (concepts) are language nouns and verbs only. It preserves words hierarchies and their semantic constraints and ontological relations among each other. Therefore, the output graph is not complex, and not huge, but rich, because each concept has its own linguistic and semantic attributes and relations that can be deduced from the analyzed input text. This prototype is the implementation for previous proposed model [5].

3 RSG GENERATION MODEL

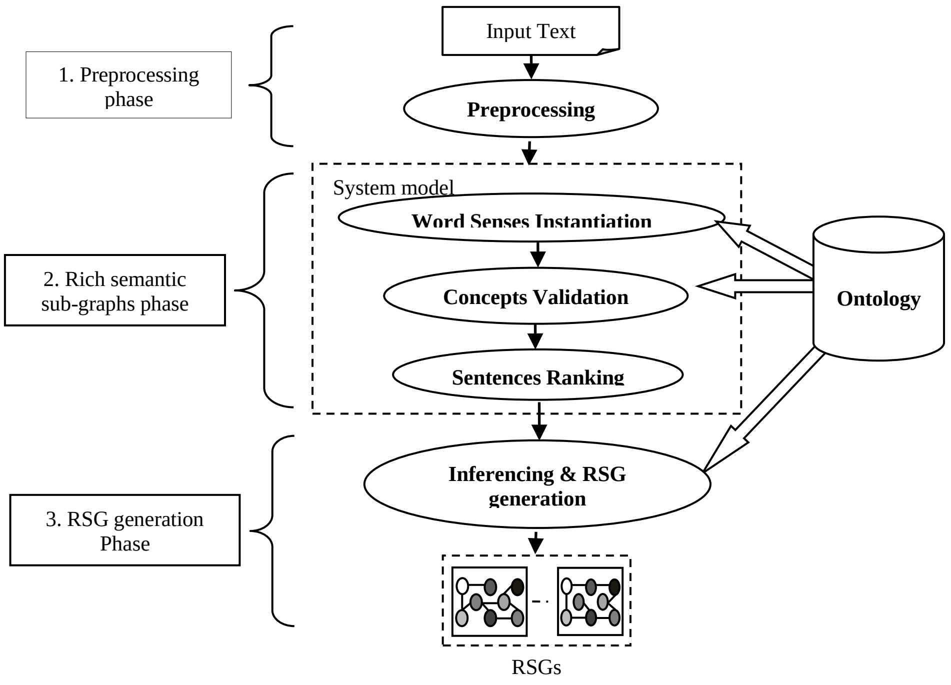


Figure 1: RSG Generating Model Architecture

Fig. 1 illustrates the overall model architecture of the proposed model [5]. It shows four main components: the input text, the system model, the ontology, and the rich semantic graph. The model ontology represents natural language concepts in a hierarchical structure. Each concept has its own attributes and relations which describe its semantics and syntax constraints. Also, inference rules are applied on the whole ontology to reason out more advanced representations. In this ontology, Wordnet lexicon has been exploited [18], [19]. As the case with any natural language, any word may have more than one sense (meaning). Therefore, there is a word concept to each sense. The word senses are classified according to their attributes which are chosen in a way to handle various semantics and linguistics issues. The rich semantic graph generation model involves three phases:

- Preprocessing phase - It is responsible to accept the input text, and converts it to preprocessed sentences. This phase includes one module called "*Preprocessing Module*", which consists of four main steps: named entity recognition, morphological and syntactic analysis, cross-reference resolution, and pronominal resolution. The main objective of this phase is to resolve the syntactic ambiguity and retrieve both set of tags (syntactic and morphological). It also retrieves typed dependency relations between words for the input text.
- Rich semantic sub-graphs generation phase - It is responsible to transform the input preprocessed sentences to set of ranked rich semantic sub-graphs. This phase includes three modules which are repeated for each input preprocessed sentence:
  - "*Word Senses Instantiation*" - This module instantiates a set of word concepts for both nouns and verbs based on the used ontology. According to the input word syntactic category tag (e.g.: verb, or noun), the ontology is accessed to instantiate a concept for each word sense that is corresponding to the word syntactic category.
  - "*Concepts Validation*" – The sentence concepts instantiated in previous module are interconnected and validated through the semantic and syntactic constraints and relationships using the ontology and the input preprocessed tags.
  - "*Semantic Sentences Ranking*" - This module aims to rank and to threshold the highest ranked rich semantic sub-graphs for each sentence.
- RSG generation phase - It is responsible to generate all possible rich semantic graphs of the whole input document. It consists of a module called "*Inferencing & RSG Generation Module*" that generates all possible combinations of the sentence rich semantic sub-graphs for the input document after applying inferencing rules in the ontology. These rules conclude more similarities and advanced relations between concepts in the final representation.

4 RSG GENERATION SYSTEM PROTOTYPE

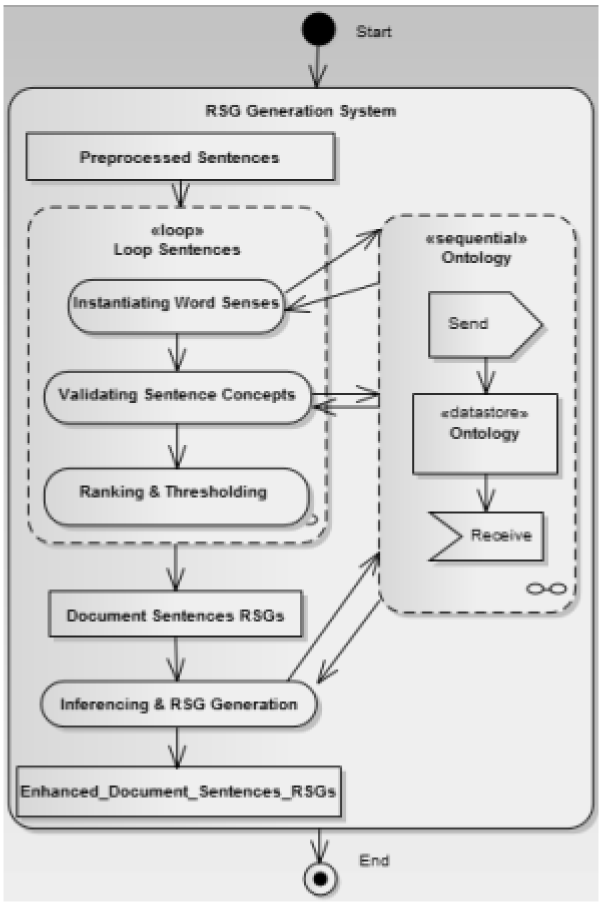


Figure 2: Overall RSG Generation System Activity Diagram

Rich Semantic Graph (RSG) system prototype is the implementation of the RSG generation model shown in figure 1. Protégé-OWL editor is used as a platform for building ontologies and knowledge-based systems with support for the Ontology Web Language (OWL). OWL is a standard ontology language to generate a common and unified text representation that can be used by different natural language applications [20], [23]. Fig. 2 shows an overall activity diagram of the RSG generation system prototype. The input is the document sentences preprocessed. Subsequently, each preprocessed sentence passes through three modules: Instantiating, validating, and ranking modules respectively. In the instantiating module, sentence concepts are instantiated from the system's ontology, and the sentence concepts are combined and validated to form different sentence rich semantic sub-graphs in the validating module. Then, the sentence rich semantic sub-graphs are ranked and threshold to get most semantically accepted sub-graphs in the ranking module. In the final stage, inference is applied to reason out the enhanced document RSG representations.

As the case with any natural language, any word may have more than one sense (meaning). Therefore, there is a word concept to each sense. The word senses are classified according to their attributes which are chosen in a way to handle various semantics and syntactic issues. One of the semantic issues is that different word meanings may lead to reason out different semantic representations. On the other hand, syntactic issues include sentences and words structures which may need special treatment and may lead to different sentence semantic representations. According to these issues, the RSG generation system uses ontology which is supplied with different rules to reserve concepts semantic and syntactic representation. Fig. 3 shows a partial noun ontology hierarchy. During RSG generation process, there are variant semantic representation cases can be met. In this system ontology, different rules and constraints are supplied to handle these cases. These cases are categorized as direct cases, indirect cases, and error detection cases. These cases are discussed briefly below:

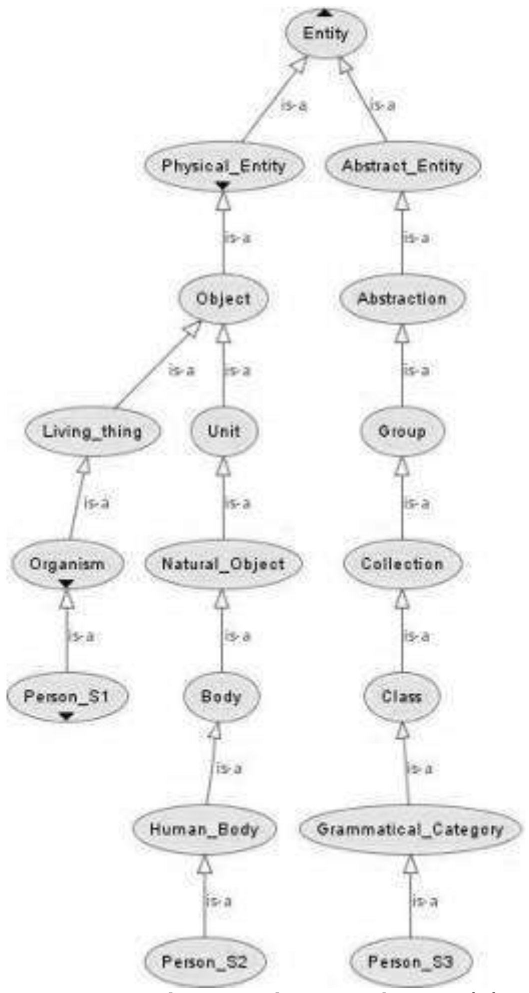


Figure 3: Partial Noun Ontology Hierarchy in Protégé

A. *Direct Cases*

In direct cases, no special action or rule needs to be applied to get RSG representation from the input sentences. In these cases, the RSG output of the system is not affected by the ontology inference rules. The sentences have direct semantic and syntactic relationships between sentence's words. For example, the sentence "*Chris eats an apple*" is a direct case. Semantically, each word in this sentence has more than one sense in the system ontology, but the highest familiar senses to each is the most accurate to relate sentence's words with each other. In the instantiation module, each noun and verb word in the example sentence instantiates its senses from the system ontology. If this word is referred in a previous sentence before, then the instantiated word senses are used, else the system ontology is searched and the word sense is instantiated. The detailed activity diagram of instantiating word senses is shown in Fig. 4.

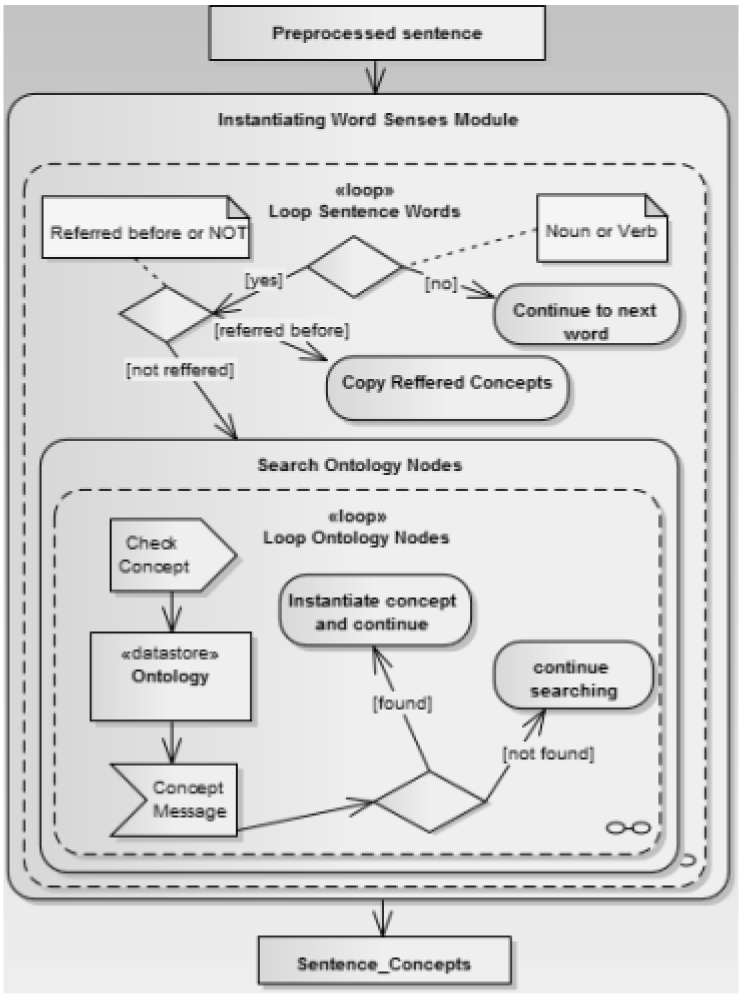
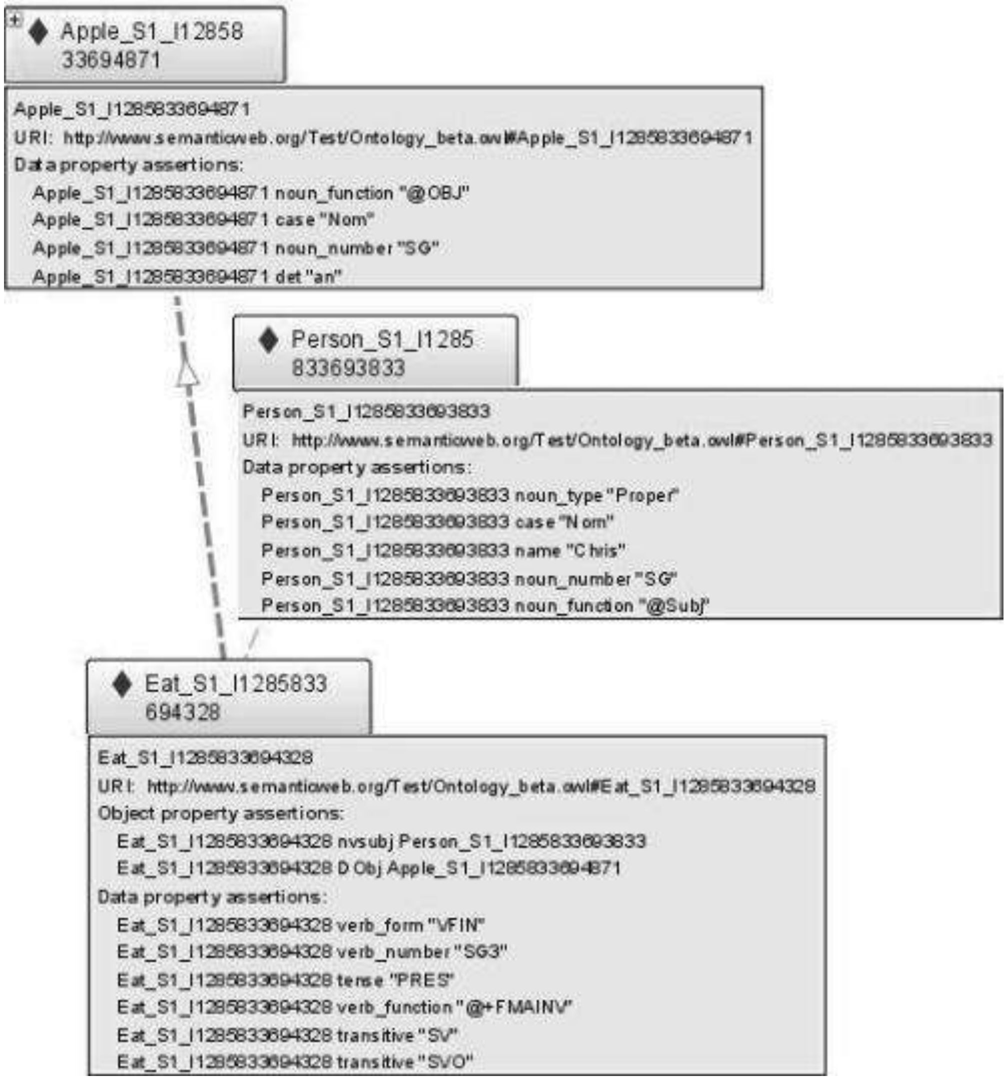


Figure 4: Detailed Activity Diagram of Instantiating Module

The output of this module is as follow: the subject "Chris" is a kind of person and it has three instances, the verb "eats" has six instances, and the object "apple" has two senses. In the validation and ranking modules, the instantiated instances are interconnected and validated to form different semantic representations to the same sentence, then the validated sentences representations are ranked and threshold to conclude most highest RSG representation. According to the output of instantiating module, there will be 36 possible semantic representations to the example sentence. After validation and ranking, the final output will be only one representation. Fig. 5 shows the final RSG output of the above sentence, where instances' names are formed of word lemma, number of sense, and automatic generated instance number. For example, "Person_S1_I1285833693833" is the first sense of word "Person" and represents the sentence's subject "Chris", "Eat_S1_I128583364328" is the first sense of word "eat" and represents the sentence's verb "eats", and "Apple_S1_I1285833694871" is the first sense of word "apple" and represents the sentence's object. At the end, the RSG system composes the sentence RSG directly without any inference rules.


Apple_S1_I12858 33694871

Apple_S1_I1285833694871
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Apple_S1_I1285833694871
Data property assertions:
Apple_S1_I1285833694871 noun_function "@OBJ"
Apple_S1_I1285833694871 case "Nom"
Apple_S1_I1285833694871 noun_number "SG"
Apple_S1_I1285833694871 det "an"

Person_S1_I1285 833693833

Person_S1_I1285833693833
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Person_S1_I1285833693833
Data property assertions:
Person_S1_I1285833693833 noun_type "Proper"
Person_S1_I1285833693833 case "Nom"
Person_S1_I1285833693833 name "Chris"
Person_S1_I1285833693833 noun_number "SG"
Person_S1_I1285833693833 noun_function "@Subj"

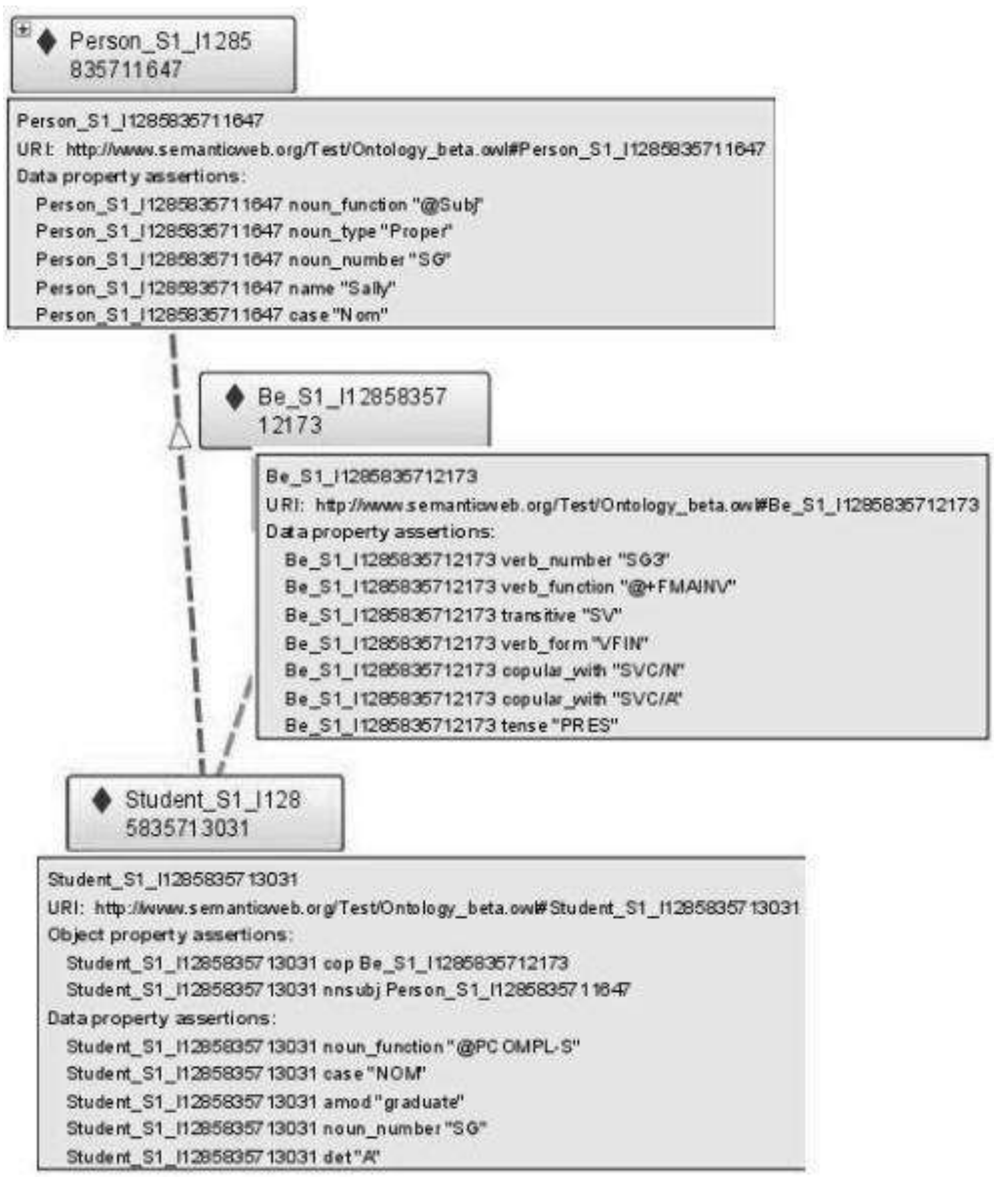
Eat_S1_I1285833 694328

Eat_S1_I1285833694328
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Eat_S1_I1285833694328
Object property assertions:
Eat_S1_I1285833694328 nvsubj Person_S1_I1285833693833
Eat_S1_I1285833694328 DObj Apple_S1_I1285833694871
Data property assertions:
Eat_S1_I1285833694328 verb_form "VFIN"
Eat_S1_I1285833694328 verb_number "SG3"
Eat_S1_I1285833694328 tense "PRES"
Eat_S1_I1285833694328 verb_function "@+FMAINV"
Eat_S1_I1285833694328 transitive "SV"
Eat_S1_I1285833694328 transitive "SVO"

Figure 5: Sentence "Chris eats an apple" Final RSG

B. Indirect Cases

These cases represent sentences where concepts are related with more complicated semantics and syntactic relations than direct cases. In these cases, the output of RSG system is affected by ontology inference rules. These rules are applied to detect similarities between concepts, so redundant relations and concepts are deleted and merged. For example, the sentence "Chris is a graduate Student" is an indirect case, where it includes the copular verb "is". The copular verbs are verbs that connect the subject to their complements, which are adjectives or nouns for the subject. In this sentence, the verb "is" links between noun to noun, where the subject is "Chris" and its complement is "student". After passing the sentence through the rich semantic sub-graph phase, the intermediate sentence RSG is generated as shown in Fig. 6. This RSG needs to be enhanced by applying inference rules. Fig. 7 shows a sample of the copular verb inference rules. Finally, the RSG generation phase will apply "*Rule 1*" to infer an enhanced RSG representation as shown in Fig. 8. It shows that "Chris" and "student" are similar instances. Therefore, these concepts have been merged in the final RSG representation output as shown in Fig. 9.



Person_S1_I1285
835711647

Person_S1_I1285835711647
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Person_S1_I1285835711647
Data property assertions:
Person_S1_I1285835711647 noun_function "@Subj"
Person_S1_I1285835711647 noun_type "Proper"
Person_S1_I1285835711647 noun_number "SG"
Person_S1_I1285835711647 name "Sally"
Person_S1_I1285835711647 case "Nom"

Be_S1_I12858357
12173

Be_S1_I1285835712173
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Be_S1_I1285835712173
Data property assertions:
Be_S1_I1285835712173 verb_number "SG3"
Be_S1_I1285835712173 verb_function "@+FMAINV"
Be_S1_I1285835712173 transitive "SV"
Be_S1_I1285835712173 verb_form "VFIN"
Be_S1_I1285835712173 copular_with "SVC/N"
Be_S1_I1285835712173 copular_with "SVC/A"
Be_S1_I1285835712173 tense "PRES"

Student_S1_I128
5835713031

Student_S1_I1285835713031
URI: http://www.semanticweb.org/Test/Ontology_beta.owl#Student_S1_I1285835713031
Object property assertions:
Student_S1_I1285835713031 cop Be_S1_I1285835712173
Student_S1_I1285835713031 nnsubj Person_S1_I1285835711647
Data property assertions:
Student_S1_I1285835713031 noun_function "@PCOMPL-S"
Student_S1_I1285835713031 case "NOM"
Student_S1_I1285835713031 amod "graduate"
Student_S1_I1285835713031 noun_number "SG"
Student_S1_I1285835713031 det "A"

Figure 6: The "Chris is a graduate student" RSG Output before applying Inference

Rule 1: " If some noun (x), and some noun (y), and some copluar verb (z), and y is subclass x, and y is subject to x, and y has copular z then x is same as y".
Rule 2: "If some noun (x), and some copular verb (z), and some thing (y), and y is copular z, and y is adjective subject to x then x has adjective y"

Figure 7: Sample of Copular Verb "Be" Inference Rules in Ontology

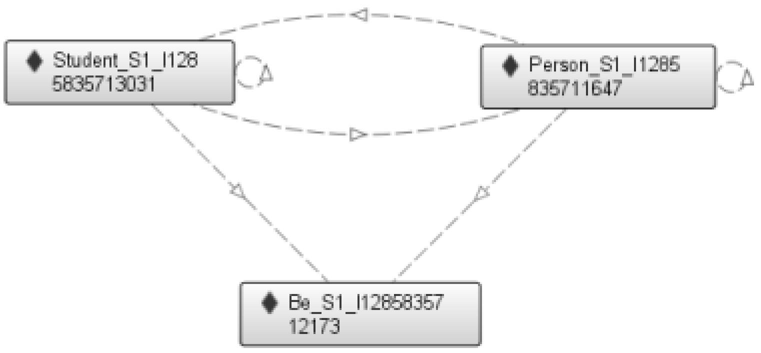


Figure 8: Intermediate RSG of the "Chris is a graduate student" Sentence

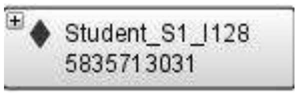
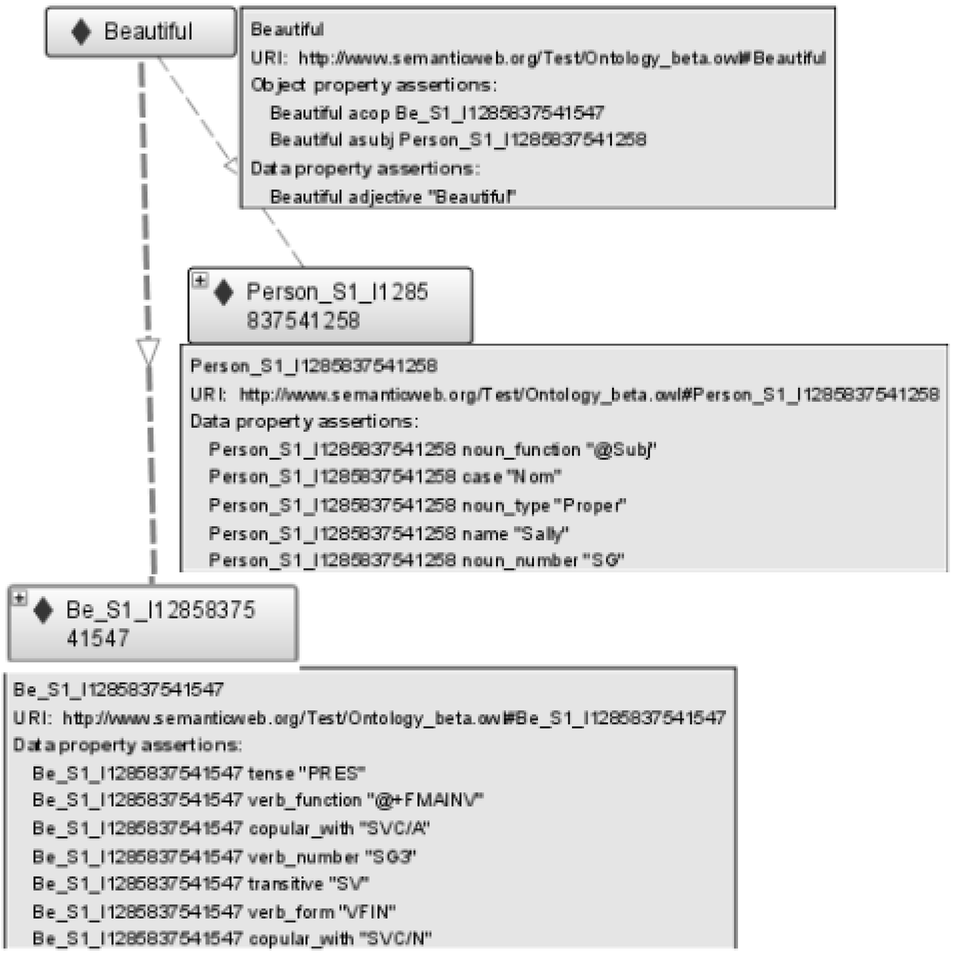


Figure 9: Final RSG of the "Chris is a graduate student" sentence

The sentence "Sally is beautiful" is another indirect case. The verb "is" links the subject "Sally" and its adjective "beautiful". Firstly, if direct case is applied, the output will be like the shown Fig. 10. Then, "*Rule 2*" in Fig. 7 needs to be applied to improve representation output (see Fig. 11). Finally, the final RSG graph will be like the shown Fig. 12.



```
Beautiful
URI:  http://www.semanticweb.org/Test/Ontology_beta.owl#Beautiful
Object property assertions:
   Beautiful acop Be_S1_I1285837541547
   Beautiful asubj Person_S1_I1285837541258
Data property assertions:
   Beautiful adjective "Beautiful"
```

```
Person_S1_I1285837541258
URI:  http://www.semanticweb.org/Test/Ontology_beta.owl#Person_S1_I1285837541258
Data property assertions:
   Person_S1_I1285837541258 noun_function "@Subj"
   Person_S1_I1285837541258 case "Nom"
   Person_S1_I1285837541258 noun_type "Proper"
   Person_S1_I1285837541258 name "Sally"
   Person_S1_I1285837541258 noun_number "SG"
```

```
Be_S1_I1285837541547
URI:  http://www.semanticweb.org/Test/Ontology_beta.owl#Be_S1_I1285837541547
Data property assertions:
   Be_S1_I1285837541547 tense "PRES"
   Be_S1_I1285837541547 verb_function "@+FMAINV"
   Be_S1_I1285837541547 copular_with "SVC/A"
   Be_S1_I1285837541547 verb_number "SG3"
   Be_S1_I1285837541547 transitive "SV"
   Be_S1_I1285837541547 verb_form "VFIN"
   Be_S1_I1285837541547 copular_with "SVC/N"
```

Figure 10: RSG Output of the "Sally is beautiful" sentence

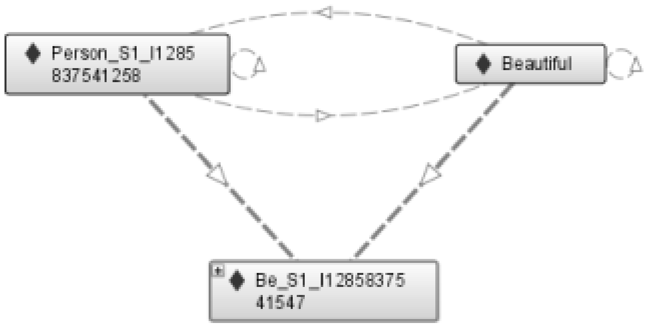


Figure 11: Inference Rule 2 output of the "Sally is beautiful" sentence

Figure 12: Final RSG of Sentence "Sally is beautiful"

C. Error Detection Cases

C.A.1 Syntactically Rejected Cases

The error detection cases have an empty list or error message output. This message describes that there is no available RSG output. In syntactically rejected cases, the parser and syntactic analyzer reject the sentence, and then the input to RSG system will be an empty list. This list means that a syntactic error found and the representation must overcome. For example, "*Chris eating apple*", this sentence has a syntactic error in verb "*eating*", and then parser sends an empty message to the system. The system handles it by displaying this message "No RSG available for the input system sentence".

C.A.2 Rejected and Low Semantically Cases

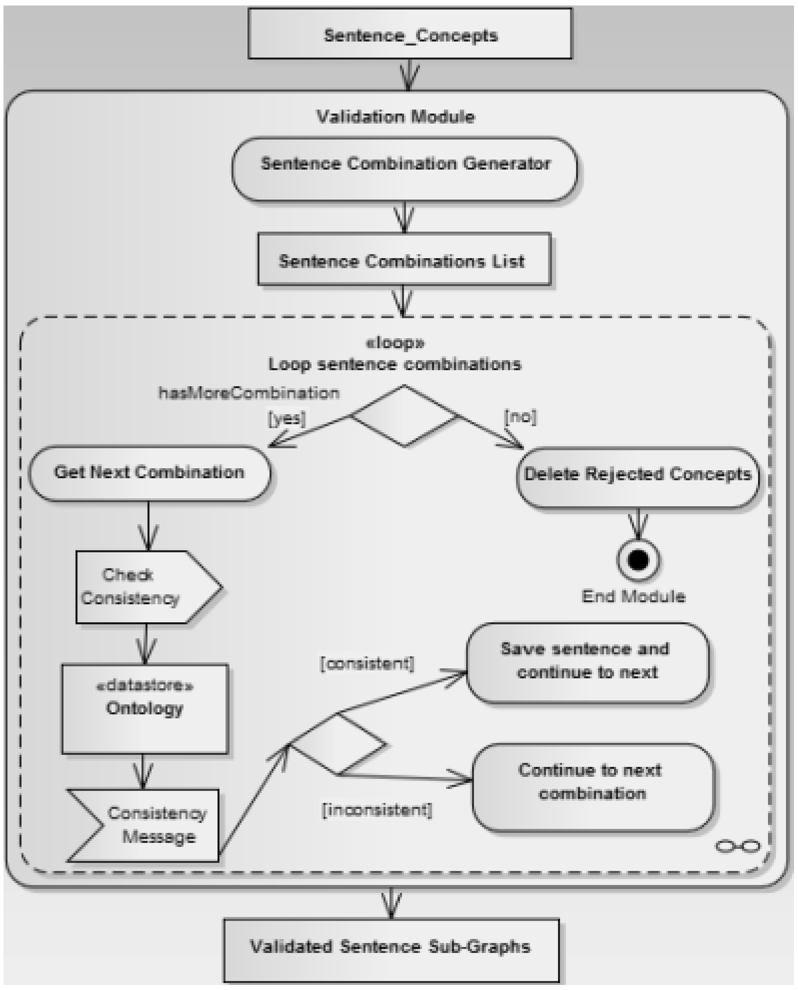


Figure 13: Detailed Activity Diagram of the Validation Module

These cases include unaccepted and weakly related semantic input sentences. The system ontology states some ontological rules and constraints on word senses and linking properties and attributes, these rules preserve the correct semantic relations between sentences' concepts. If the input sentence is rejected or low semantically related, it will be rejected in the validation module due to absence of acceptable relations between sentence's senses, where the instantiated sentence concepts are combined to form sentence rich semantic sub-graphs. For each concept combination, the system ontology is checked to confirm combination consistency. The inconsistent combination is rejected. The validation module detailed activity diagram is shown in Fig. 13. On the other hand, the input sentence may also be rejected in ranking module due to its low rank value. For example, "Chris eats wood", "Chris" is a kind of person which has three senses, "eats" is a verb and has six different senses, and "wood" is a noun and has eight different senses. Finally, there will be 144 different combinations between the sentence's concepts. Some of these combinations can be rejected in validation phase or in ranking phase. In validation phase, if verb "eats" means to swallow and noun "wood" means a plant material, it will be rejected because "eats" object must be an edible food. On the other hand, if verb "eats" means to use up or to exhaust and "wood" means a musical instrument, it will be rejected due to low semantically relatedness. The RSG system shows the log of each representation combination, it displays accepted and rejected combinations and rejection reasons.

5 CONCLUSION

This paper presented Rich Semantic Graph (RSG) generation system prototype. It transformed the input text to a graph called "*Rich Semantic Graph*" (RSG). In RSG, verbs and nouns of a document are represented as nodes along with edges corresponding to semantic and linguistic relations between them. "Rich Semantic Graph" (RSG) is a common rich semantic representation to be exploited in different natural language processing applications such as report abstraction and machine translation. The RSG has the following features:

- Each node corresponds to a concept verb or noun in the input text.
- Each node is associated with semantic and linguistic issues which reserves concepts' meaning.
- Each node has additional attributes which are supplied from the input preprocessing tags.
- Relations between nodes are semantic or syntactic connections between concepts to reserve text coherence and sequence.

The RSG generation system prototype discusses different cases with some examples. These examples reflect some natural language ambiguity and the solutions are suggested to overcome it. The RSG system has the following capabilities:

- The RSG represents simple and unambiguous semantic texts without any special rule or constraint.
- The RSG uses inferencing rules and constraints to reason out more acceptable text semantic representations.
- The RSG detects and rejects unsemantically related or low semantically related concepts.

On the other hand, the RSG generation system has some limitations:

- Some input sentences haven't any output due to small semantic rank values. If some sentence's concept replaces with any of its synonym and this synonym has better rank value, then some representation can be deduced to this sentence better than nothing.
- Some RSG sentences output need to be smaller or condensed with reserving context meaning. It can be achieved through more semantic inferencing rules.
- Other input sentences haven't any output due to syntactic errors. These errors can be autocorrected, and then some RSG representation output can be reached.

These limitations may be considered as the following future work.

REFERENCES

[1] Mostafa Aref, "A Multi-Agent System for Natural Language Understanding", *Proceedings of the International Conference on Integration of Knowledge Intensive Multi-Agent Systems*, 1-3, Cambridge MA, PP 36-40, 2003.

[2] Elaine Rich, Kevin Knight, *"Artificial Intelligence"*, Second Edition, McGrawHill Inc, Chapter 4-11, 1991.

[3] Robert Stevens, Carole A. Goble and Sean Bechhofe, "Ontology-based Knowledge Representation for Bioinformatics", *IBM Systems Journal*, volume 40, issue 2, February 2001.

[4] Ibrahim F. Moawad & Mostafa Aref, "A Semantic Graph Reduction Approach for Abstractive Text Summarization", *the Ninth Conference on Language Engineering*, Cairo, Egypt, December 6-7, 2009,

[5] Ibrahim F. Moawad, Mostafa Aref, Soha Said, "ONTOLOGY-BASED MODEL FOR GENERATING TEXT SEMANTIC REPRESENTATION", *to be published in International Journal on Intelligent Computing and Information Sciences (IJCICIS)*, Faculty of Computer and Information Sciences, Ain Shams University, Cairo, Egypt.

[6] D. Rusu, B. Fortuna, M. Grobelnik, D. Mladenić , "SEMANTIC GRAPHS DERIVED FROM TRIPLETS WITH APPLICATION IN DOCUMENT SUMMARIZATION", *an international journal of Computing and Informatics* , Volume 33, Number 3, 2009.

[7] Lorand Dali, Delia Rusu, Blaž Fortuna, Dunja Mladenić, Marko Grobelnik, "Question Answering Based on Semantic Graphs", *Semantic Search 2009 Workshop, 18th International World Wide Web Conference WWW2009*, Madrid, Spain, April 21st, 2009.

[8] Jure Leskovec, Marko Grobelnik and Natasa Milic-Frayling , "Extracting Summary Sentences Based on the Document Semantic Graph", Microsoft Research, Technical Report No. MSR-TR-2005-07, January 2005.

[9] Jure Leskovec, Marko Grobelnik and Natasa Milic-Frayling, "Impact of Linguistic Analysis on the Semantic Graph Coverage and Learning of Document Extracts", *Proceeding of the 20th National Conference on Artificial Intelligence (AAAI)*, vol. 3, Pittsburgh, Pennsylvania, 2005.

[10] Jure Leskovec, Marko Grobelnik and Natasa Milic-Frayling, "Learning Semantic Sub-graphs for Document Summarization", *Proceedings of the 7th International Multi-Conference INFORMATION SOCIETY IS*, Jozef Stefan Institute, Ljubljana, Slovenia, Volume B, 2004.

[11] Jure Leskovec, Marko Grobelnik and Natasa Milic-Frayling, "Learning Sub-structures of Document Semantic Graphs for Document Summarization*", Workshop on Link Analysis and Group Detection (LinkKDD), 2004.*

[12] M. Aref, "Object-Oriented Approach For Morphological Analysis", *Proceeding of the 15th National Computer Conference*, Dhahran, Saudi Arabia, pp. 5-11, 1997.

[13] Mostafa Aref, "Object Orientation in Natural Language Processing", *Proceedings of the 13th International Conference on IEA/AIE 2000*, New Orleans, PP. 591-600, June 19-22, 2000.

[14] UNDL Foundation, "*The Universal Networking Language (UNL) Specifications*", Version 3, Edition 3, UNL Center, UNDL Foundation, Geneva, Switzerland, December 2004.

[15] Luis iraola ,"Using Wordnet for linking UWs to the UNL UW System", *International conference on the convergence of knowledge, culture, language and information technologies*, December 2- 6, 2003.

[16] Dr. Waralak V. Siricharoen, "Ontologies and Object models in Object Oriented Software Engineering", *IAENG International Journal of Computer Science*, 33:1, IJCS_33_1_4, 2005.

[17] Lisiane Goffaux and Robert Mathonet, "A technique for customizing object-oriented knowledge representation systems, with an application to network problem management", *Proceedings of the Eleventh International Joint Conference*, vol. 1, August 20-25, 1989.

[18] Fellbaum, C." *WordNet: An Electronic Lexical Database*", MIT Press, May 1998.

[19] Miller,G., Beckwith,R., Fellbaum,C., Gross,D., and Miller,K, *Five Papers on WordNet.* Cognitive Science Laboratory, Princeton University, Princeton, 1990.

[20] Michael K. Smith, Chris Welty, Deborah L. McGuinness (10 February 2004), "OWL Web Ontology Language Guide", W3C Recommendation, Available from: www.w3.org/TR/owl-guide, (accesses 1 November 2010)

[21] UNL Specification Web Site: http://www.undl.org/unlsys/unl/unl2005/, (accessed 1 November 2010)

[22] UNL Universal Words Ontology Web Site: http://www.undl.org/unlsys/uw/unlontology.htm, (accessed 1 November 2010)

[23] Protégé Web Site: http://protege.stanford.edu, (accessed 1 November 2010)

ON THE MODELING OF NON-KEYWORD INTERVALS FOR SPOKEN-TERM DETECTION IN ARABIC

M. Hesham[*1], M. F. Abu-EL-Yazeed[**], and A. Toulan[**]
[*]*Engineering Math. & Physics Dept., Faculty of Engineering, Cairo University*
[1*] mhesham@eng.cu.edu.eg
[**] *Electronics & Communications Engineering Dept.,*
Faculty of Engineering, Cairo University, 12613, Egypt

***Abstract*— There are two approaches followed in searching/indexing an audio stream, namely; key word spotting and spoken term detection. First one is the keyword spotting approach based on searching through the acoustical-features domain while the other is the spoken-term detection (STD) where the searching is through string domain. The task of spoken term detection is defined as being a two-stage process in which the audio stream is, firstly, indexed according to word or sub-word units (e.g. phonemes), and then search is performed over the indexed audio. One major problem of searching, through audio streams, is the presence of unpredictable or unexpected words in incoming utterances. For resolving this problem, a direct approach is to split the training data into keyword and non-keyword data. The keywords are represented by models trained using the keyword speech, and the non-keyword models are trained using non-keyword data. Two problems arise in this approach that it may overly detect keywords, or misrecognize non-keyword in the sentence as keywords. In order to overcome these problems, we propose a method for modeling the non-keyword intervals or out-of-vocabulary (OOV) for Arabic. In the proposed method, the OOV models are defined based on pattern structure in Arabic phonology.**

Firstly, a hidden-Markov modeling (HMM) approach is used in the training of an automatic-speech recognition (ASR) engine for modern standard Arabic (MSA). The phoneme-based ASR engine is, then, employed in the keyword spotting system. The acoustic features of the pattern structure of Arabic is exploited in a distributed a model to cover non-keyword intervals in a task-independent manner. The distributed structure models are used to capture out-of-vocabulary words belonging to all possible patterns in Arabic, while keyword models are built based on exact phonemic structure of significant words or words combination. This improves the system's ability to detect non-keyword vocabularies independently. Additional class of fillers associated with different word classes reduces false alarm of keyword detection.

The system is trained to detect about 20 Arabic keywords using both techniques; keyword spotting and STD approaches. The keyword spotting approach is built based on HMM models concatenation for phonemes while the STD employed 1-best lattice-search approach. The obtained results for the spotting system, achieves 90% precision rate for keyword spotting approach while the recall rate is smaller. The other approach of spoken-term detection reached the 80% in recalling and smaller rate for precision.

Keywords: Keyword Spotting, Spoken-Term Detection, Arabic Speech Recognition, Hidden Markov Model (HMM).

1 INTRODUCTION

The Searching for a keyword within a speech utterance is one of important applications of speech recognition technology. Ability to perform such searching/indexing task, together with typical automatic speech recognition for speech utterances whose underlying word sequences follow fully-defined syntaxes, effectively is crucial to the development of human-machine spoken dialog systems. There are two approaches followed in searching/indexing an audio stream, namely; key word spotting and spoken term detection, as discussed recently in [1][2][3] and [4]. One major problem with using speech recognizer for delivering sequence of words to other modules in spoken dialog systems is its robustness [3]. Another problem is the presence of unpredictable or unexpected words in incoming utterances. Therefore, a speech recognizer must cover a large number of vocabularies, as well as grammars, in order to support every sentence possibly spoken by the users. This also includes un-precedently-used words and non-speech sounds, such as fillers, that are rather common in spoken languages.

The techniques of key-word searching try to detect of all occurrences of any given word in a speech signal. Trying to uncover these keywords, it may overly detect keywords, or misrecognize non-keyword in the sentence as keywords. The misrecognition may be called false alarm. One factor contributing to a high false alarm rate is that there are some non-keyword elements (or may be called out-of-vocabulary(OOV)) that have similar pronunciation to some of the keywords in the system. These elements may confuse the keyword spotting system to confirm the presence of a keyword, even though none of them exist[3].

Although several approaches exist for modeling of keyword spotting system, the most common ones are based on HMMs. In such approaches, a set of HMMs (nonkeywords or filler models) is chosen to represent the OOV intervals and another one for the keywords [4]. The performance of an HMM-based keyword spotter strongly depends on the ability of the OOV models to represent non-speech intervals without rejecting the correct keywords (false rejections). Therefore, the choice of an appropriate OOV model set is a critical issue. A survey on the topic can be found in [4]. The most common approaches are as follows:

- The training corpus for a specific task is split into keyword and non-keyword (extraneous) data. The keywords are represented by HMMs trained using the keyword speech and the OOV models are trained using the extraneous speech. The main disadvantage of such approaches is the task dependence. Model retraining is required when the vocabulary changes. Moreover, the training data must include a large number of keyword occurrences to achieve robust training.
- The OOV models are selected from a set of common acoustic models. In this case, a speech corpus for a separate task is used to train only one common acoustic set. A subset of this set is used for OOV models. A typical case is to represent the keywords by context-dependent HMMs, and the non-keyword portions by context-independent HMMs. In some works, a subset of context-dependent models is also used as OOV models. The main disadvantage of such methods is the high rate of false rejections (percentage of keywords rejected). The higher local likelihood of the OOV models causes the OOV entries to be decoded instead of keywords. Most keyword spotters use two sets of acoustic models trained using keyword and non-keyword speech. The performance of a keyword spotter may be increased by training phonemic OOV models using a large corpus of non-keyword speech [4]
- Modeling the non-keyword intervals based on the use of bilingual HMMs. In this case, the OOV models may be trained using a speech corpus of a language other than the target one. This can introduce a task-independent keyword spotter, and to overcome the problem of the high rate of false rejections[4].
- The work in [3] illustrates the use of acoustic modeling of three different structures, including syllables, fillers and keywords. Filler models and syllable models are applied to capture OOV words, while keyword models extract significant words from speech utterances. Filler models associated with syllable models reduce false alarm of keyword detection. Three kinds of filler models were described in [3].

As most keyword spotters use a set of Hidden Markov Models (HMM) for their components (keywords and non-keywords), many HMM parameters still need to be re-estimated and tested for Arabic speech via different applications.

On the other hand, the task of spoken term detection is defined as being a two-stage process in which the audio stream is, firstly, indexed according to word or sub-word units (e.g. phonemes), and then search is performed over the indexed audio. The indexing may be as a 1-best string, or as an N-best lattice. Lattice-based methods offer significantly faster search, as the speech is processed just once by a speech-recognition engine. Some works adopt the approach of searching for terms in the output of a large-vocabulary speech recognition (LVCSR) system, though a common finding is that these approaches yield high miss rates (i.e., low recall). Hybrid methods based on the combination of keyword-spotting (which gives high recall) and sub-word lattice search have proven successful in combining the strengths of both methods [1].

In this work, a new model is defined for building linguistic structures which can cover an efficient OOV set for Arabic. We propose a method for modeling the non-keyword intervals or (OOV) for Arabic. In the proposed method, the OOV models are defined based on pattern structure in Arabic phonology.

The HMM phoneme models along with the proposed OOV structure are used to build a task-independent system for spoken-term detection of Arabic. Experiments are conducted through an audio content of ELRA database [5]. The results are, then, compared to elicit the efficacy of both searching approaches and the proposed OOV structure for Arabic.

2 PATTERN STRUCTURE OF ARABIC MORPHOLOGY

Arabic is marked by a limited vocalic system and a rich consonantal system. There are typically three basic vowels /a i u/ which are attested in both their short and long forms [6] ch1. Arabic operates by what is known as the root and pattern system [7]. The root is a semantic abstraction consisting of two, three or (less commonly) four consonants from which words are derived through their superimposition of template patterns. Roots may be used in association with a particular vowel pattern which determines phonological structure and specifies lexical and syntactic function.

Basic noun and verb stems in Arabic comprise a consonantal root and pattern. The pattern can be further divided into two elements: a prosodic template and a vocalic melody. [6]. The consonantal root is always fully independent of the prosodic template, the vocalic melody by contrast shows independence for relatively few morphological categories. For verbal noun, the prosodic template CVCVVC, and the vocalic melody i-a, comprises the combination of both [8]. These separate out the linguistic information carried by the word pattern into two components labeled the vocalic melody – the sequence of vowels specified by the word pattern – and the CV-Skeleton – the overall abstract pattern of consonants (C) and vowels (V) that it also specifies. The vocalic melody conveys syntactic meaning such as voice (active/passive). The CV-Skeleton contributes a rich variety of other syntactic information, as well as specifying the phonological shape of the word. [9].

The root-and-pattern morphology of Arabic is most commonly described using examples of derivational verbs. One form of trilateral verb has the stem pattern CVCCVC with a geminate middle radical. In standard Arabic, the imperfect is formed by changing the quality of the right most stem vowel and adding an imperfect prefix. The passive is formed by a change in the vocalic melody [6].

Another example, one of these patterns may take the form $C_1aC_2aC_3VC_4$ where C_1, C_2, C_3 represent three consonants of the root in relation to the sounding vowel. Many patterns are the result of a series of derivations. Accordingly, some of these patterns are, totally, predictable, and if the form does not already exist, will be given predictable meaning when coined [6].

In general, Arabic has a limited number of patterns which are pronounced according to similar acoustical configuration (parameters). This pronunciation follows the same phonological pattern which has close resemblance to acoustical features of the utterance. Specifically, we can propose that they share same spectral characteristics. This property results in some sort of clustering in vector-quantization space of related features. The mel-frequency cepstarl coefficients (MFCC) are the used features in most of speech recognition algorithms [10]. These feature are of spectral-type, accordingly, we expect that the words, sharing the same pattern, may easily tend to group to the same cluster in feature space. This fact is also attested by the conceptual basis of MFCC which, closely, manifests the source-excitation state of vowel production. This concept can help in proposing a comprehensive model for OOV. This pattern-based model can be a better candidate for the minimum OOV set of words due to a limited number of patterns in Arabic.

3 A Distributive Model of OOV

The problem in defining OOV models is the trade-off between false alarm and miss-detection rates. Using OOV models with less acoustic details introduces high false alarm rate, while using models with more acoustic details raises the miss-detection rate. A survey on this topic was presented in [4].

In this work, the non-keyword domain is divided into two main classes namely; non-keywords and fillers. Each class is, then, divided in distributive subdomains. Each subdomain is restricted by closer acoustic features. This distribution is proposed to solve the problem of biased decision on nonkeyword class. This condition is ensured by choosing closer pronunciation of the elements of any subdomain. The pronunciation condition is based on pattern structure of Arabic –discussed in the previous section- for non-keyword subdmoain.

Another subclass; filler models are acoustic models trained on phonemes that is not repeated frequently in Arabic words. These phonemes are used, even, if they were included in the targeted keywords. Filler elements also include words that are rarely to be included in keywords like prepositions. This was done based on speech data in our training set.

In the proposed model, two approaches are considered for choosing the nonkeyword elements; syllabic and shuffled nonkeyword models.

A. *Pattern-nonkeyword Model*

The pattern-nonkeyword model is built for the patterns known in Arabic. We used many words as examples for each pattern. As mentioned above, any word is composed of two parts; the first part corresponds to a root consisting of two or three syllables. The other part is the pattern or vocalic melody. They are aggregated in about 16 classes representing, approximately, most of pattern structures which are frequently used in Arabic. This ensures the distributive nature of the model over the language domain.

B. *Shuffled-nonkeyword Model*

The other nonkeyword model is based on replacing the starting-consonant of syllable each word defined in the previous section. These words did not have any meaning within Arabic. We denote them as having shuffled structure since we shuffle the consonants through the used words. The inherent distribution is proposed to solve the problem of biased decision on nonkeyword class. The keyword, the non-keyword and filler models are constructed from the acoustic models of phonemes.

Examples on the shuffled nonkeyword are shown in Table I. The syllable /$a/ is replaced by /Fa/ and, then, the new word has no meaning in Arabic. The idea is to construct OOV model from meaningless words that are acoustically different from the keywords. This property prevents false alarms when the pronunciations of two words are close to each other.

TABLE I Shuffled nonkeyword Examples

Nonkeywords of one class	Meaning in English	Arabic Transcription	Shuffled version with no meaning	Arabic Transcription
$axS	Person	شخص	FaxS	ثخص
$axSan	A person	شخصا	FaxSan	ثخصا
$axSy	personal	شخصي	FaxSy	ثخصي

* The words are transcribed in Roman of ELRA [5].

4 KEY WORD SPOTTING

Several approaches exist for modeling both keywords and non keywords in a keyword spotting, the most common ones are based on HMMs. Keyword spotting systems generally use HMMs of phone-based sub-word units [11]. In such approaches, a set of HMMs is chosen to represent both the keywords and the non-keyword intervals [4]. In this keyword spotter, the keywords and OOV units are modeled through a concatenation of monophone HMMs [12]. The system usually consists of several steps. The first step is to train an HMM based speech recognition engine. Secondly, a dictionary is built for the keywords models and the OOV models. OOV words are built the same way as the keyword models, but are used to represent non-keywords in our recognition network. Finally, to detect the keywords in an audio stream, the file is passed through the speech recognizer. The output should define which words were detected and the places where they were detected in the stream.

The speech recognition network for this method consists of a number of paths from the beginning node to the end node. Each of the paths passes through a model corresponding to a sound segment from the set of all possible non-keywords, possible keywords, and the set of all possible outputs from the filler models [3]. This network is shown in Fig. 2. In this network, *$nonkw_i$* means one of the non-keyword classes. Also, *fil_i* represents a filler model while *KW_i* is the model of *i^{th}* element.

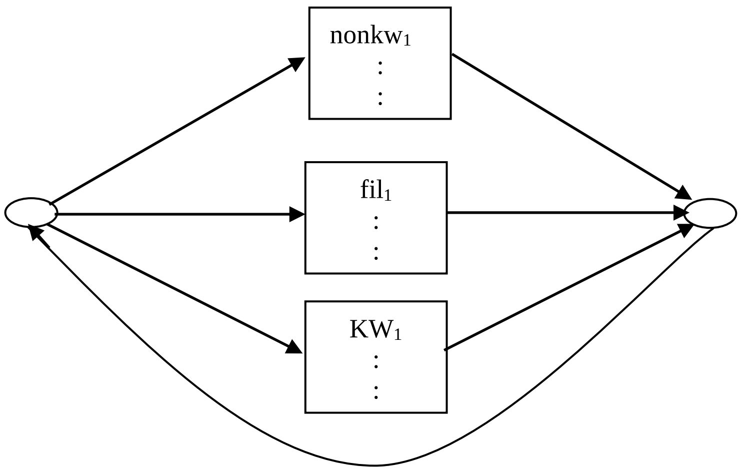


Fig. 2 Keyword spotting network.

5 SPOKEN TERM DETECTION

For the spoken term detection system [1], lattice-search approach is used. The lattice-search system provides the flexibility that we don't need to refer to the original audio streams each time the streams are searched, which can increase the system speed when searching large amount of data.

In N-best lattice search, Viterbi algorithm [12] is known to provide all path fragments that matches a certain model. The method is based on confidence measure calculation for a certain keyword from a Viterbi network. Equation (1) illustrates how the confidence measure is calculated,

$$C_{KW}^{phn} = L_{alpha}^{phn}(KW) + L_{phn}(KW) + L_{beta}^{phn}(KW) - L_{best}^{phn} \qquad (1)$$

where $L_{phn}(KW)$ is the log likelihood of the keyword, computed from the summation of the acoustic log likelihood of its constituent phonemes. $L_{alpha}^{phn}(KW)$ is the forward likelihood of the best path from the beginning of the lattice to the keyword. $L_{beta}^{phn}(KW)$ is the backward likelihood from the end of lattice to the keyword. L_{best}^{phn} is the 1-best likelihood path over the complete lattice. In 1-best lattice search method [1], also called lexical access. It involves the calculation of costs for recognizer errors. The distance between two strings is defined as the minimum number of error transformations used to derive one from the other [17] when the error transformations are defined in terms of substitution, deletion and insertions errors as defined by Levenshtein metric [18]. A threshold on the confidence score is calculated on a training portion of the database to give the required system performance [1]. We consider another error type for continuation as in [1] beside deletion, insertion and substitution errors. Continuation error simply indicates that the character inserted is the same as the character preceding it. Levenshtein distance is, then, calculated between the recognized string and targeted keyword string using *Dynamic Programming Algorithm*. Dynamic programming solves small, constrained versions of the problem. When the constraints are tight, the function is simple to compute, and then the constraints are systematically relaxed until finally they yield the value of the desired answer [19]. The algorithm uses an effective yet rather brute-force approach that essentially looks at every possible way of transforming the source string to the target string to find the least number of changes.

Later in the testing phase we use these costs as defined in equation (2) to spot the actual keywords. We start by selecting a small window referred to as W_k^{min}, calculate the total cost from this window. The window cost is based on individual errors costs calculated previously from the training phase. Then we grow linearly by a single phoneme and redo the process until we reach W_k^{max} then we shift by one phoneme and repeat till the whole utterance is finished. We remove any overlaps, by selecting the window with the least cost between intersecting windows. Finally, we decide whether or not we detected a keyword by a threshold, also calculated from the training phase. We repeat this procedure for every keyword defined in our dictionary.

$$C_{sub} = -\log\frac{N_{sub}(h,k)}{N_{tot}(h,k)}, \qquad C_{ins} = -\log\frac{N_{ins}(h,k)}{N_{tot}(h,k)},$$

$$C_{del} = -\log\frac{N_{delb}(h,k)}{N_{tot}(h,k)} \quad \text{and} \quad C_{con} = -\log\frac{N_{con}(h,k)}{N_{tot}(h,k)} \qquad (2)$$

Where C_{sub} is the subsititution cost and N_{sub}(h, k) is defined as the total number of substitutions of test symbol k for reference symbol h, N_{ins} (h, k) is defined as the total number of insertion of test symbol k for reference symbol h, N_{del} (h) is the total number of deletions of reference symbol h, N_{con} (h, k) is the total continuation of test symbol k for reference symbol h and N_{tot} is the total number of occurrences of reference symbol h.

The dynamic programming algorithm is used again to calculate the overall cost of matching keyword against recognizer output.

For example, consider the following recognizer output *EwzirHFax....* So we will start with W_k^{min} =3 and end with W_k^{max} =8. So we will have the following windows sequence [Ewz, Ewzi, Ewzir, EwzirH, EwzirHF, EwzirHFa, wzi, wzir, wzirH, ...]

The cost comparison of 2 of these windows is shown in the tables TABLE II and TABLE III

TABLE II Costs of W_k^{min} window

Searched Keyword		w	z
	0	0.6310	0.9335
w	0.4662	0	1.3998
a	0.8023	0.3361	1.7359
z	1.5839	1.1176	0.3361
E	2.0794	1.6131	0.8316
r	2.6218	2.1555	1.3740

TABLE III Costs of W_k^{max} window

Searched Keyword		w	z	i	r
	0	0.6310	0.9335	2.0938	2.6674
w	0.4662	0	1.3998	2.5601	3.1337
a	0.8023	0.3361	1.7359	2.7086	3.4698
z	1.5839	1.1176	0.3361	1.9898	4.1663
E	2.0794	1.6131	0.8316	1.2854	3.4216
r	2.6218	2.1555	1.3740	1.8278	1.2854

So the cost for the window *wz* is 1.3740 and for the window *wzir* is 1.2854 . The window *wzir* will be considered as the window with minimum cost and if it's below the threshold calculated on the training database it will be considered a match or spotted.

6 EXPERIMENTS AND RESULTS:

The automatic speech recognition (ASR) engine is trained using an Arabic Globalphone database from ELRA [5]. The provided database has approximately 4908 speech sentences from about 84 speakers. The recording sampling frequency is 16 ks/s with 16 bits quantization.

The database is divided into two subsets; one for training and the remaining part is preserved for testing. The training set is composed of about 3069 recorded waves. Each recording is associated with an Arabic transcription composed from about 35 phonemes. The HTK toolkit [12] is used. The frame size is 25 ms with overlapping of 15 ms. The feature vector is taken of length 39 including 12 mel frequency cepstrum coefficients (MFCC), and the zero MFCC coefficient, which is proportional to the total energy in the frame and their corresponding delta and acceleration coefficients. Each phoneme is modeled with 3 states with 16 gauss-mixture model (GMM). The accuracy of phoneme recognition approaches the order of 65%. A word recognizer is built based on concatenations of phoneme HMM models.

The keyword searching is conducted through the structures described in section II. The first one is the syllabic non-keyword (OOV) model. The second structure has both syllabic OOV with interchanged and shuffled consonant in syllables. The third considers different variants of keyword transcription besides the shuffled syllables in OOV word classes.

According to pattern structure and variations in pronouncing vocalic melody of different words, we used more than one transcription for each keyword. Examples on keyword variants are listed in TABLE IV according to different pronunciations of vowels according to neighboring phonemes.

Table IV KEY-WORD variants

Keyword	الاتحاد الأوروبي
Meaning in English	European Union
Variants of Keyword transcription	CilCitiHAd CilCUrUbE
	CalCitiHAd CilCUrUbE
	CalCittiHAd lCalCUrUbE
	CilCitiHAd lCalCUrUbE
	CalCitiHAd CalCUrUbE
	CalCittiHAd CalCUrUbbE

TABLE V KEY-WORD SEARCHING FOR 20 KWS

CONDITION	RECALL	PRECISION
SYLLABIC OOV MODEL FOR NONKEYWORDS	70.39%	86.9%
SHUFFLED OOV MODEL FOR NONKEYWORDS	64.25%	82.14%
SHUFFLED OOV MODEL CONTAINED IN ONE SET (NONDISTRIBUTED)	45.25%	93.10%
SHUFFLED OOV MODEL AND KW VARIANTS	74.3%	82.61%
SYLLABIC OOV MODEL AND KW VARIANTS	75.42%	83.85%

The results presented in TABLE V are summarized from many experiments using sufficiently-repeated words through the used database. The results shows the effect of replacing the consonant of original words used in non-keyword models, these results ensure that the vocalic melody is not sufficient to model the OOV. Despite that the shuffling of syllables in OOV model did not, significantly, reduce the false alarms, it will help for more generic spotting system.

For spoken term detection system using Lattice search method, the correct detection depends on the ability to correct recognizer output. To do that we calculate all the recognizer errors on a portion of the database, associate costs to different errors, store them and consider this as the training phase. This portion should be big enough to represent all expected errors from the system. We use the 3069 utterances of our speech database for the training purpose [5]. HResults is a tool provided by the HTK toolkit [12]. It provides a good implementation for Levenshtein distance. It mainly calculates insertion, substitution and deletion errors, but no continuation errors. So, a modified version of HResults was made to provide all the results we need from Levenshtein algorithm.

For the spoken term detection system, we, applied the 1-best lattice search approach. In this trial, we used a general rule for window lengths, and the rule was applied on all keywords. TABLE VI shows the results from spoken term detection experiment where W_k^{min} = $N_{K/2}$ + 1, N_K is defined as the number of phonemes in dictionary entry for keyword K, and W_k^{max} = W_k^{min} + N_K and W_k^{min} and W_k^{max} defined for each keyword where N_k is the length of keyword k. The recall rate was 79.33% and the precision rate was 64.84%. After trying several experiments we chose specific window lengths for each keyword, the recall rate reached 75.98% and precision rate reached 75.14%. We also experienced a great enhancement in system speed by reducing the number of used windows.

TABLE VI STD searching for 20 KWs

Condition	Recall	Precision
Fixed widow length	79.33%	64.84%
Variable window length	76.54%	75.69%

7 DISCUSSIONS AND CONCLUSIONS

This work considers the problem of keyword searching through Arabic speech. Two approaches are tested through this work for keyword spotting in Arabic. First one is the keyword spotting approach based on searching through the acoustical-features domain while the other is the spoken-term detection where the searching is through string domain. One essential problem in building this searching system is the definition of out-of-vocabulary (OOV) words beside the searched keywords. To solve this problem for Arabic, the pattern structure of Arabic is exploited in building a model for OOV part of the system. The OOV model is chosen acoustically homogenous and distributed over a wide range of frequently used words in Arabic. The elements of OOV were chosen in a distributive manner according to pattern structure of Arabic.

The proposed OOV model is tested in two versions; one of them is based on words covering most of Arabic patterns which are limited. The other is called shuffled nonkeyword model in which the consonants are interchanged between training non-keywords. This concept is partially tested due to lack of data. Also, we include different transcriptions for a keyword as pronounced in practice. This model of keyword-spotting helps in improving the spotting accuracy. The obtained results for the proposed systems, achieve more than 90% precision rate for keyword spotting approach while the recall rate is smaller. The other approach of spoken-term detection approaches 80% in recalling and smaller rate for precision.

The obtained results show the efficacy of employing the pattern structure of Arabic in defining OOV words. Also it is proved that hybrid methods based on the combination of keyword-spotting and sub-word lattice search is needed for Arabic as reported for other languages [1].

Acknowledgement

The authors would like to thank ITIDA, Egypt (Information Technology Industry Development Agency) for its support via provided fund. They are, also, fully indebted for the valuable discussions and help provided by Prof. Dr. Waleed Fakhr, Arab Academy of Sciences & Maritime.

References:

[1] J. Tejedor, D. Wang, J. Frankel, S. King and J. Colas, “ A comparison of grapheme and phoneme-based unitsfor Spanish spoken term detection”, speech comm, 50 (2008) 980–991.

[2] J. Keshet, D. Grangier, S. Bengio, “Discriminative keyword spotting”, Speech Communication 51 (2009) 317–329

[3] S. Tangruamsub, P. Punyabukkana, and A. Suchato, “Thai speech keyword spotting using Heterogeneous acoustic modeling”, Research innovation and Vision for the future, IEEE International Conference, 2007.

[4] P. Heracleous , T. Shimizu, “ A novel approach for modeling non-keyword intervals in a keyword spotter exploiting acoustic similarities of languages”, Speech Communication 45 (2005) 373–386.

[5] ELRA, ELRA-S0193, GlobalPhone Arabic, ELDA S.A., 2006.

[6] J. C. E. Watson, The Phonology and Morphology of Arabic, Oxford university press Inc., NewYork, ch.6, 2002.

[7] M. C. Bateson, Arabic Language Handbook, Gorgetown university press, Washington D. C., ch.1, 2003. Washington D. C., ch.1, 2003.

[8] K. Brown and S. Ogilive, (edited by), Concise Encyclopedia of Languages, J. C. E., Watson, Arabic as an Inflecting Language, Elssevier, pp. 431-434, 2006.

[9] S. Boudelaa, and W. D. Marslen-Wilson, “Abstract morphemes and lexical representation: the CV-Skeleton in Arabic”, Cognition, International Journal of Cognitive Science, vol. 92, pp. 271–303, 2004.

[10] J. Benesty, M. Sondhi, and Y. Huang (EDs), Springer Handbook of Speech Processing, Springer Berlin-Heidelberg, 2008.

[11]Y. Ling, “Keyword spotting in continuous speech utterances,” Master’s thesis, McGill University, Montreal, 1999.

[12] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, “The HTK Book” (for HTK Version 4.30), Cambridge University Engineering Department, England, 2006.

[13] K. Schulz and S. Mihov, “Fast string correction with levenshtein-automata,” International Journal of Document Analysis and Recognition, vol. 5, pp. 67–85, 2002.

[14] Z. Su, B.-R. Ahn, K.-Y. Eom, M.-K. Kang, J.-P. Kim, and M.-K. Kim, “Plagiarism detection using the levenshtein distance and smith-waterman algorithm,” in Proceedings of the 2008 3rd International Conference on Innovative Computing Information and Control (ICICIC ’08), (Washington, DC, USA), p. 569, IEEE Computer Society, 2008.

[15] A. A. Kanso, “An efficient cryptosystem delta for stream cipher applications,” Comput. Electr. Eng., vol. 35, no. 1, pp. 126–140, 2009.

[16] P. Caballero-Gil and A. F´uster-Sabater, “A simple attack on some clock controlled generators,” Comput. Math. Appl., vol. 58, no. 1, pp. 179–188, 2009.

[17] A. V. Aho and T. G. Peterson, “A minimum distance error-correcting parser for context-free languages,” SIAM Journal on Computing, vol. 1, no. 4, pp. 305–312, 1972.

[18] V. I. Levenshtein, “Binary codes capable of correcting deletions, insertions, and reversals,” Soviet Physics Doklady, vol. 10, no. 8, pp. 707–710, 1966.

[19] G. Heineman, G. Pollice, and S. Selkow, *Algorithms in a Nutshell*. O’Reilly, 1st. ed., 2009.

Designing and Implementing Arabic Text-To-Speech (ArTTS)

Hassanin M. Al-Barhamtoshy[*1], Fahd Al-Hiedary[*], Mansour Al-Johany[*] and Wajdi H. Al-Jedaibi[*]

Faculty of Computing and Information Technology, King Abdulaziz University, SA

[1]hassanin@kau.edu.sa

***Abstract* - This article presents a new approach to analyze mixed-text that includes Arabic words and numeral texts. Therefore, Arabic digits and numbers pronunciation included in this article with the associated related rules. Also, the different rules of Arabic graphemes pronounced letters are presented, designed and implemented in ArTTS.**

ArTTS concatenated these pronounced graphemes and produced sounded Arabic words. The result that produced from numbers matches the Arabic pronouncing of numbers. The result produced from letters is accepted and understood to the listener. ArTTS deals with about 90 rules of numbers and diactrize rules.

This approach centers on a modular, mixed-lingual morphological and syntactic analyzer, which additionally provides accurate language identification on morpheme level, word and sentence boundary identification in mixed-lingual texts. This approach can also be applied to word identification in languages without a designated word boundary.

1 INTRODUCION

Speech refers to the processes associated with the production and perception of sounds used in spoken language [1]. In recent years, the massive development of information and technology provides a special technique for speech that is used in applications and used by the computer to implement some of the tasks that need to be voted.

Speech technology relates to the technologies designed to duplicate and respond to the human voice. They have many uses: to aid the voice-disabled, to communicate with computers without a keyboard, to market goods or services by telephone and to enhance game software [2]. Speech technology uses several processes that study speech signals and the processing methods of these signals. These signals are usually processed in a digital representation, whereby speech processing can be seen as the intersection of digital signal processing and natural language processing.

1.1. Speech Recognition (SR)

Speech recognition, also known as automatic speech recognition or computer speech recognition [3]. Where the computer utilizes audio input for entering data rather than a keyboard. Speaking into a microphone, for example, produces the same result as typing words manually with a keyboard. Simply stated, voice recognition software is designed with an internal database of recognizable words or phrases. The program matches the audio signature of speech with corresponding entries in the database. Though turning speech into text might sound easy, it is an extremely difficult task [4]. The problem lies in the virtually infinite array of individual speech patterns and accents, compounded by the natural human tendency to run words together. Some Speech recognition systems, called discrete speech systems, require the user to speak clearly, slowly and to separate words [4]. Speech recognition is especially useful in business where it can replace a live operator to funnel calls, disseminate information, take orders and perform other highly useful functions.

1.2. Text To Speech (TTS)

TTS, or Text-To-Speech, is the digitized audio rendering of computer text into speech, synthesized speech through computer's speakers; Figure (1). TTS programs can be useful for a variety of applications. For example, proofreading with TTS allows the author to catch awkward phrases, missing words or pacing problems. TTS can also convert text files into audio MP3 files that can then be transferred to a portable MP3 player or CD-ROM [5]. This can save time by allowing the user to listen to reports or background materials in bed or while performing other tasks [5]. The quality of a speech synthesizer is judged by its similarity to the human voice, and by its ability to be understood.

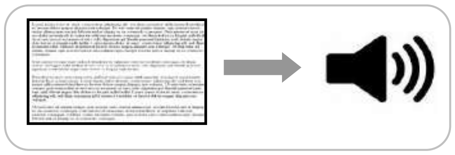


Figure 1: Text To Speech (TTS) Digitizing

1.3. Speech Properties

1.3.1. Phoneme

In human language, a phoneme is the smallest posited linguistically distinctive unit of sound. Phonemes carry no semantic content themselves [6, 7]. An example of a phoneme is the /ق / sound in the words (قارب) and (قيعان). (In transcription, phonemes are placed between slashes). Even though, most native speakers don't notice the different of speech [6].
An example of different phoneme that would cause a change in meaning, producing words like (عرب) (substituting /ع/), (غرب) (substituting /غ/). In many languages, each letter in the spelling system represents one phoneme [6, 8].

1.3.2. Allophone

Allophone is one of several similar speech sounds (phones) that belong to the same phoneme [9]. An example of allophone /ت/ and /ط/ sound in the words (متابعة) and (مطالعة), the two letters have different phoneme but they have the same sounds. Changing the allophone won't change the meaning of a word, the result may sound non-native [7, 8, 9].

1.3.3. Morpheme

Morpheme is the smallest linguistic unit that has semantic meaning. In spoken language, morphemes are composed of phonemes and in written language, morphemes are composed of graphemes (the smallest units of written language) [9, 10]. The word "unbreakable" has three morphemes: "un-", a bound morpheme; "break", a free morpheme; and "-able", a bound morpheme. Different morphs representing the same morpheme are Allomorph [7, 10].

1.3.4. Syllable

Syllable is defined as a unit of organization for a sequence of speech sounds. Syllables are often considered the phonological (building blocks) of words [7, 11]. An example of syllable is the word (قواعد) the two syllables here are (قوا) and (عد), Figure (2).

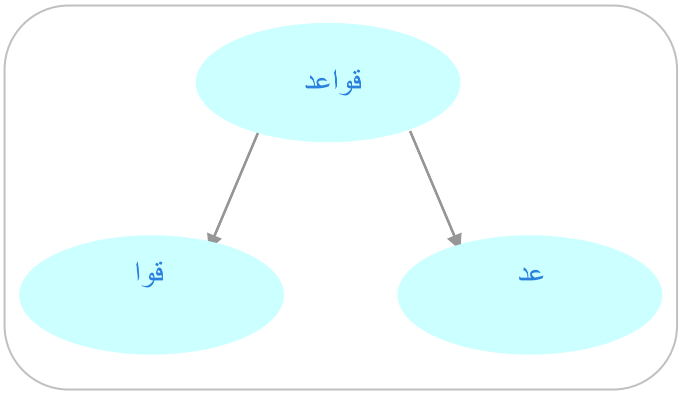


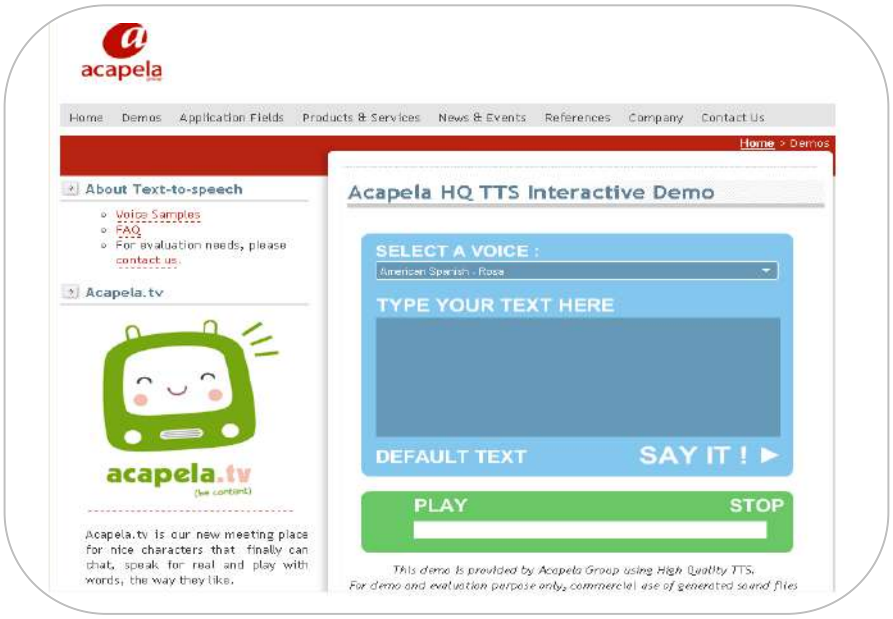
Figure 2: Example of Arabic Syllable

1.4. Research Scope and Background

The scope of the research project converts Arabic text to speech, by analyzing the input text and using special characteristics of Arabic language, generate Arabic phonemes, match between the Arabic phonemes and the stored ones in the computer, then produce digitized audio of Arabic text. Consequently, a study of Arabic grammars and properties of speech will be included.
TTS has many benefits, including scientific and journalized texts, automated reader, telephone response, searching in web, online audio player, music and media player.

1.4.1. Acapela TTS Program

The first model is provided by Acapela Group using high quality TTS [12]. The Acapela voice solutions speak 25 languages using over 50 characters voices. It allows any company to acquire its own exclusive voice at an affordable price. See Figure (3).



For evaluation needs, please contact us.

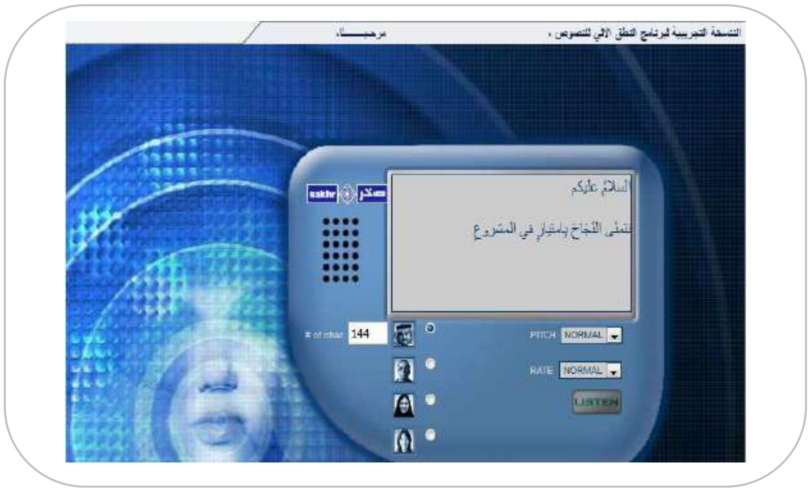
Acapela.tv is our new meeting place for nice characters that finally can chat, speak for real and play with words, the way they like.

This demo is provided by Acapela Group using High Quality TTS.
For demo and evaluation purpose only, commercial use of generated sound files

Figure 3: Acapela TTS demo [12]

1.4.2. Sakhr TTS Program

The second model is presented by Sakhr TTS [13]. Such demo works on Arabic language only. The special characters of Arabic language like (ً_, ُ_, ْ_) are used (Figure 4). The model has two male voices and two female voices. Such program has different pitch.



النسخة التجريبية لبرنامج النطق الآلي للنصوص ، مرحبـــــا،

السلام عليكم

نتمنى النجاح بامتياز في المشروع

# of char 144

PITCH NORMAL

RATE NORMAL

Figure 4: Sakhr TTS demo [13]

Most attempts to provide text-to-speech for Modern Standard Arabic (MSA) have concentrated on solving the problem of diacritic assignment. The paper [14] described an approach to the task of generating speech from MSA text, which provides the information required for imposing an appropriate intonation contour in Arabic text.

Voice conversion, i.e., modification of a speech signal to sound as if spoken by a different speaker, finds its use in speech synthesis with a new voice without the necessity of a new database. The paper of [15] introduces two new simple non-linear methods of frequency scale mapping for transformation of voice characteristics between male and female or childish. The frequency scale mapping methods were developed primarily for use in Czech and Slovak text-to-speech (TTS) system designed for the blind and based on the Pocket PC device platform. It uses cepstral description of the diphone speech inventory of the male speaker, using the harmonic speech model. Three new diphone speech inventories corresponding to female, childish and young male voices are created from the original male speech inventory. Listening tests are used for evaluating voice transformation and quality of synthetic speech [15].

Text-to-speech synthesis systems have to deal with texts containing inclusions of multiple other languages in the form of phrases, words or parts of words in multilingual countries. In such multilingual cultural settings, listeners expect a high-quality text-to-speech synthesis system to read such texts in a way that the origin of the inclusions is heard. The challenge for a text analysis component of a text-to-speech synthesis system is to derive, from mixed lingual sentences, the correct polyglot phone sequence and all information necessary to generate natural sounding polyglot prosody [16]. The article [16] presents a new approach to analyze mixed-lingual sentences. This approach centers on a modular, mixed-lingual morphological and syntactic analyzer, which additionally provides accurate language identification on morpheme level and word and sentence boundary identification in mixed-lingual texts. To date, this mixed-lingual text analysis supports any mixture of English, French, German, Italian and Spanish. Because of its modular design, it is easily extensible to additional languages.

2 ArTTS ANALYSES AND ORGANIZATION

2.1. Object-Oriented Analysis and Design - UML Diagrams

UML consists of nine different diagram types, each focused on a different way to analyze and define the system. These diagrams are summarized briefly here [18]:

- Use Case Diagrams show the externally visible behavior of the system.
- Activity Diagrams show an elaboration of the behavior of the system.
- Component Diagrams show architecture of the system.
- Sequence Diagrams show object interactions over time.
- Collaboration Diagrams show object interactions with emphasis on relations between objects.
- Class Diagrams show class definition and relations.
- Statechart Diagrams show state changes in response to events.
- Deployment Diagrams show physical architecture of the system.
- Package Diagrams show the design hierarchical.

2.2. Requirement Scenario of ArTTS Design Screen

1. Type a text into the "Text box"
2. Press on "Speak" button to listen to the text words.
3. Copy a text from anywhere and paste it in the "Text box".
4. Clear the "Text box".
5. Cut a text from anywhere and paste it in the "Text box".
6. Import a text file and export the text.

2.3. Use Case Model

Figures (5) and (6) show the use case model and screen design of the ArTTS. User deals with the system by using User Screen, which includes many operations of the ArTTS.

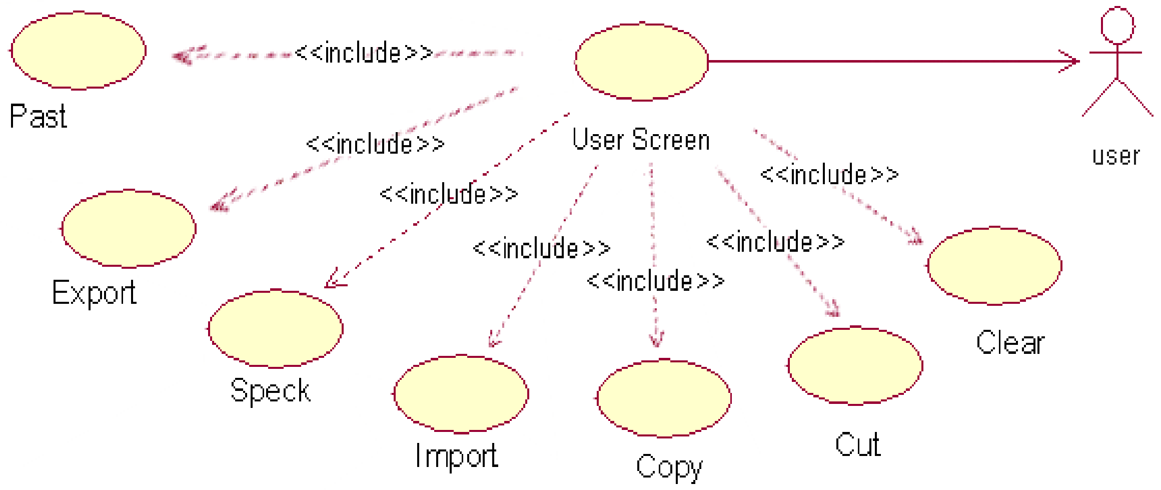


Figure 5: ArTTS Use Case Diagram

2.4. Description of "User Screen" Use Case

1. "User Screen" is created and displays the following screen

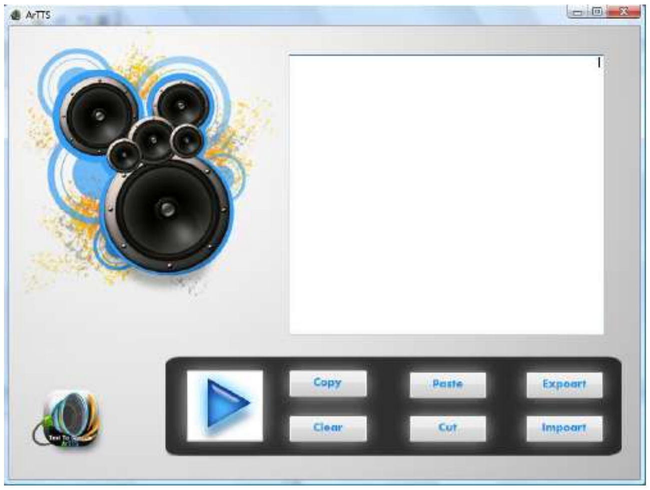


Figure 6: ArTTS Screen Design

2. If (Speak) button is selected, text in text box will be split into words.
   2.1. Each word will be split into graphemes.
   2.2. If grapheme is a letter, each letter in Arabic format will call its phoneme path from "Phoneme Path Table" that has the following structure:
   2.3 If grapheme is a number, each number will call its number path from "Numbers Path Table" that has the following structure:
   2.4. ArTTS will play the paths- stream of phonemes (the process of speak button).
3. If (Copy) button is selected, the system will make a copy of the selected text in computer memory.
4. If (Paste) button is selected, the copied text in memory will be pasted in the "TEXT BOX", where the pointer is.
5. If (Clear) button is selected, all text in "TEXT BOX" will be selected and deleted.
6. If (Export) button is selected, the file will be saved in the selected location as text format.
7. If (Import) button is selected, the imported text file will be shown in the "TEXT BOX".

2.5. Analysis Model

The analysis model is specified as an initial object structure in the form of class and object diagrams, which is able to realize the behavior specified in the use cases. Therefore, many UML diagrams are created to describe the proposed system and the main interface of ArTTS.

Sequence diagram is the most popular UML artifact for dynamic modeling, which focuses on identifying the behavior within the proposed system. Such diagram describes the user sequence diagram of the ArTTS.

3. STRUCTURE OF ArTTS ENGINE

The proposed ArTTS engine is a model that synchronizes Arabic speech output by:

1. Breaking down Arabic text into orthographic words, by the scanner/parser module.
2. The orthographic words are analyzed and then broken down into phonemes, using the following sub-steps:
   a. Analyze the input of numerals of text that require conversion to pronounced phonemes, using Arabic numeral conversion rules.
   b. Analyze the input orthography words and convert this stream into phonemes, using Arabic phonological rules.

3. Concatenate the stream of phones via concatenation module, using the builder module.
4. Generate the digital audio.

3.1. ArTTs Layout Architecture

ArTTS Engine is a concatenative engine that uses linguistic rules to generate artificial sounds. The ArTTS produces output by concatenating recordings of units of real human speech. These units are phonemes that have been extracted from larger units of recorded speech, but may also include words. Figure (7) shows the proposed engine of ArTTS model using several steps.

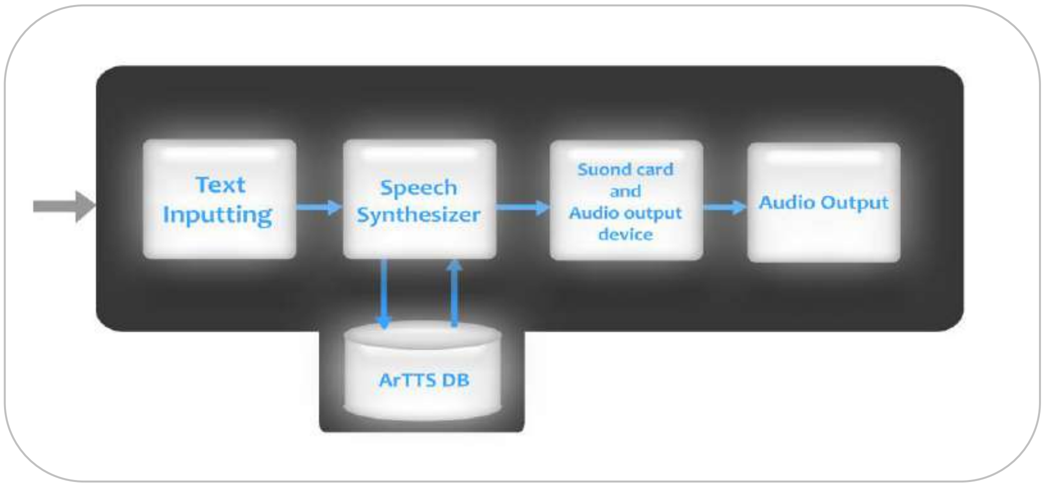


Figure 7: ArTTS Engine Structure

3.1.1. Text Input

In this step input, text box is needed to enter Arabic text. This entered text is used as input to the second step of speech synthesizer.

3.1.2. Speech Synthesizer

The description of speech synthesizer of ArTTS is illustrated in the following. Figure (11) shows 5 phases, including Scanner (Tokenizer) which contains Stream of words, Generation Rules and Builder. This step includes entire 4 phases:

Phase 1 (Scanner) or (Tokenizer)

In this phase, text will break down into words (tokens).

Phase 2 (Splitter)

The splitter will break words into graphemes.

Phase 3 (Generation Rules)

These rules are used to determine the most special cases of the pronunciation of Arabic text [8] that ensure the appropriate representation of the text with the rules of Arabic pronunciation. To do this phase, ArTTS Engine will use two sum phases each phase will use 3 sequential processes for words and numbers.

A. Diactrize Rules

1. Process Using Fags

This process is to check each rule if it will be applied or not.

2. Process Look Ahead

This process is for checking forwards for next grapheme.

3. Process Phoneme Sound Path

Connect to database to get the phonetic grapheme path.

B. Numbers Rules

1. Process Using Fags

This process is to check each rule if it will be applied or not.

2. Process Look Ahead

This process is for checking forwards for next letters.

3. Process Phoneme Sound Path Generator

Connect to database to get the phonetic letters path.

Phase 4 (Builder)

A complete phonetic word will be generated, therefore full phonetic words will be ready to be sent to the sound card. Consequently, sound card converts digital audio (sound file) to acoustical signal and amplifies through speaker. Therefore, users hear these spoken words**.**

The ArTTS system is based on the source-filter speech model or the diphone and allophones description is shown in Figure (8). Arabic text to be synthesized is entered as an input sequence of characters. It is converted to the combination of diphones and phone units, through the prosody generator. Each of these units has its collection description in the speech database.

According to Arabic language, prosody rules are applied. The resulting synthetic speech is generated pitch-synchronously. We have used the original male voice database and two derived databases with given formant modification factors according to: (1) Allophones knowledge base, (2) Diphone knowledge base and (3) mapping methods. The basic pitch frequency and the corresponding description of the speech database are selected and attached for synthesis by the choice of a voice.

The synthetic speech is generated as a sum of sine waves with given frequencies, amplitudes, and phases. The frequencies are multiples of pitch frequency, amplitudes and phases that are given by sampling the frequency response of the vocal tract model, given by the default stored voices.

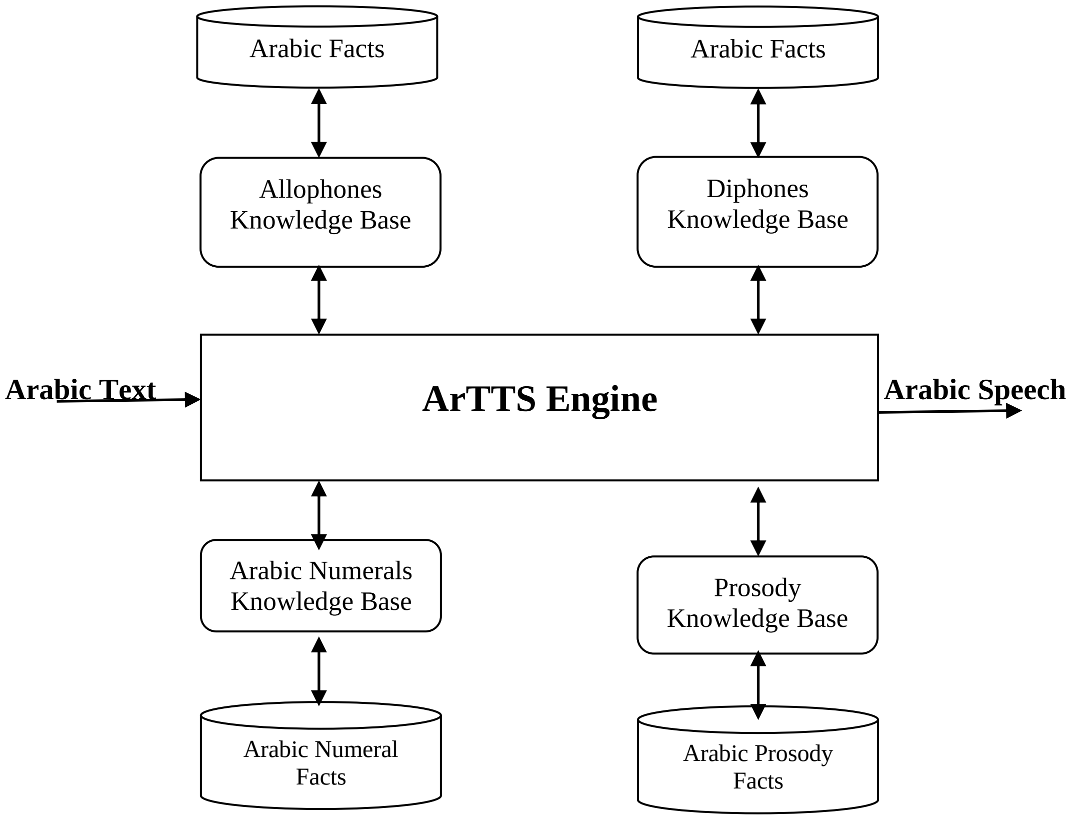


Figure 8: The Proposed Model Structure of ArTTS

This architecture of the ArTTS strictly separates language-independent algorithms from language-dependent linguistic and acoustic data. Furthermore, voice-independent and voice-dependent parts are separated. The voice-independent part includes text analysis and phonological processing. It transforms the Arabic input text into the so-called phonological representation, i.e., a minimal, voice-independent abstract description of the speech to be synthesized. On the other hand, the phonological representation includes the phonetic symbols and the abstract descript on of stream of sentence to be uttered. The voice-dependent part, composed of concatenation control and speech signal generation, produces the speech signal from phonological representation.

The ArTTS engine transforms a text paragraph by paragraph in four steps into speech. Figure (9) illustrates these steps. The applied methods of the four corresponding system components are as follows:

1. Arabic text analysis derives the morphological structure of Arabic words and delivers the phonetic transcription and language identification of each morpheme. For Arabic text analysis, strictly rule-based processing is applied, which uses word and sentence grammars and two-level rules for lexicon-to-surface mapping implemented: Allophones and Numerals knowledge bases.
2. Phonological processing applies phonological transformations, which are formulated using the so-called multi context rules. It also assigns sentence stress and phrase boundaries, based on the syntactic structure of a sentence. This abstract description, together with the phonetic transcription of each word, constitute the phonological representation.
3. Concatenation control generates, from the phonological representation, the physical stream sequence parameters. These are the stream values of all phones and the equivalent waves files of a sentence. Phone duration and wave file control are realized by means of stored models, which directly map the symbols of the phonological representation onto wave file index and fundamental frequency values.
4. Speech signal generation is based on concatenation of diphone units extracted from natural speech (files stream). Prior to concatenation, the diphones have to be modified such that they match the specified phone duration and fundamental frequency values.

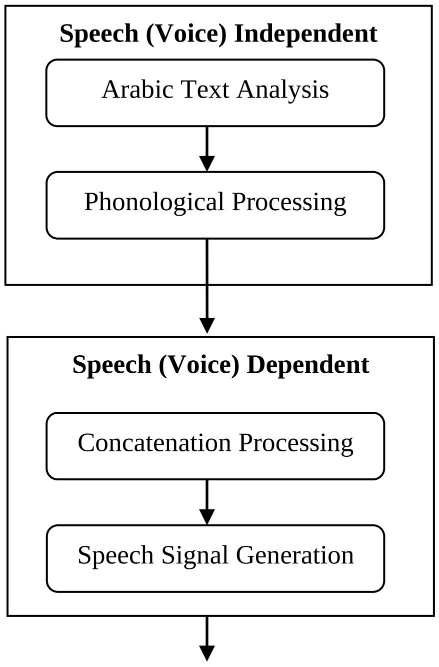


Figure 9: The Architecture of the ArTTS

3.3. ArTTS Knowledge Base

3.3.1. Look up Database

A database is used in ArTTS to store all paths of letters and numbers. To get the path of letters or numbers, ArTTS will (Look up to Database) by using two classes.

1- Letters path class: for letters paths.

2- Numbers path class: for numbers paths.

3.3.2. Numbers Path Table

The numbers Path Table contains 3 columns (ID, Number, Number Path). Although, the number path column contains the paths of sounded numbers files (.wav files). So, the goal of this table is to call the paths of numbers to play it as a sound. This table contains two sections, these sections will be described in the following.

A. Needed Sounds

Start from ID (0-9) containing the paths of the most used sounds in numbers system in Arabic language such as follow

ID	Word in English	Word in Arabic	Word pronounce in English
0	And	و	Waa
1	Ten	عشر	Ashar
2	1 hundred	مئه	Me'aa
3	1 thousand	ألف	Alf
4	thousands	آلاف	Aalaf
5	1 thousand	ألفاً	Alfan
6	2 hundreds	مئتان	Me'atan
7	2 thousands	ألفان	Alfan
8	One	احدى	Ehda
9	two	اثنا	Ethna

So, when the input text is a number, those paths will be called and stored in an array to use them directly when ArTTS needs to use these sounds.

B. Numbers Sounds

Start from ID (10 to the end of table's rows), containing the paths of sounded numbers from (1-9) in two types:

1. First Type

For Individual numbers such as (1, 2, 3)

2. Second Type

For tens, hundreds and thousands numbers, this type is divided into two modes :

2.1. First mode

Numbers needed to be spoken as individuals in a stream of numbers ("133" last 3 will be spoken "ثلاثةُ" not "ثلاثه")

2.2. Second mode

For tens numbers (20, 50, 80)

Now, all sounds of numbers and sounds of words used in Arabic numbers speech are ready to create the sound of any stream of numbers in the range of (1 – 99999). As an example, (53082) will be structured in ArTTS as follows:

ثمانون+و+اثنان+و+ألفاً+خمسون+و+ثلاثةُ (from left to right)

Thalathato waa Khamsoon Aalfan waa Ethnan waa Thamanoon.

All special cases of Arabic numbers speech are included in Numbers Rules at Speech Synthesizer level.

3.3.3. Letters Path Table

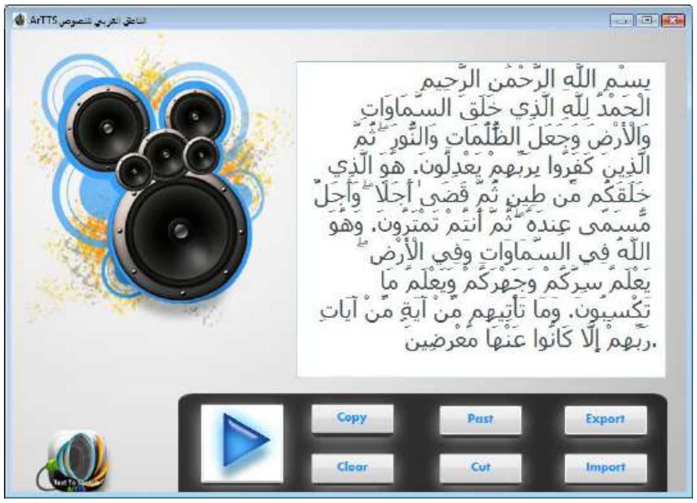
This table also includes 3 columns (ID, Letters, Path), and each letter has 8 different types of sounds (ـَ ـُ ـِ ـْ ـّ ـً ـٌ ـٍ). So, the received grapheme that is sent by (Diactrize Rules) will be compared inside the table to get the path, not inside the code. Then, the path will be sent to (Mapping Phase) to play it.

4 ArTTS IMPLEMENATION AND TESTING

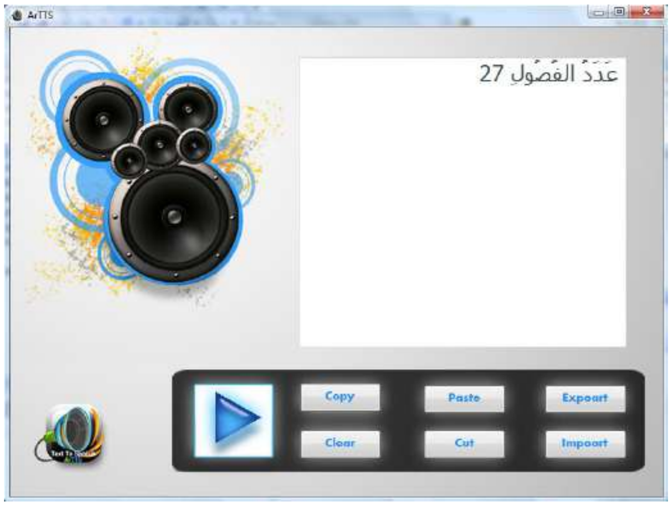
As mentioned, the ArTTS structure is composed of 4 stages. The second stage is the very important one. Therefore, the following section describes the C# code to operate with speech synthesizer mechanism.

4.1. ArTTS Implementation

C Sharpe language is used to implement ArTTS, including scanner, splitter, generation rules and builder modules. The scanner is used to break down into words (Tokens). The splitter will break words into graphemes. Figures (10-a) and (10-b) include two examples to test ArTTS.



بِسْمِ اللَّهِ الرَّحْمَٰنِ الرَّحِيمِ
الْحَمْدُ لِلَّهِ الَّذِي خَلَقَ السَّمَاوَاتِ
وَالْأَرْضَ وَجَعَلَ الظُّلُمَاتِ وَالنُّورَ ۖ ثُمَّ
الَّذِينَ كَفَرُوا بِرَبِّهِمْ يَعْدِلُونَ. هُوَ الَّذِي
خَلَقَكُمْ مِنْ طِينٍ ثُمَّ قَضَىٰ أَجَلًا ۖ وَأَجَلٌ
مُسَمًّى عِنْدَهُ ۖ ثُمَّ أَنْتُمْ تَمْتَرُونَ. وَهُوَ
اللَّهُ فِي السَّمَاوَاتِ وَفِي الْأَرْضِ ۖ
يَعْلَمُ سِرَّكُمْ وَجَهْرَكُمْ وَيَعْلَمُ مَا
تَكْسِبُونَ. وَمَا تَأْتِيهِمْ مِنْ آيَةٍ مِنْ آيَاتِ
رَبِّهِمْ إِلَّا كَانُوا عَنْهَا مُعْرِضِينَ.



عَدَدُ الفصُول 27

Figure (10-a) **Figure (10-b)**

In Figure (13-a), input text will be represented by ArTTS as follow:عَ+دَ+دُ+ +أْ+لْ+فُ+صو+لِ+ +سبعةُ+و+عشرون

5 RELATED WORKS

An ARABTTS system was judged, well acceptability and visually impaired listener has accepted the system very well [17]. The intonation lacks naturalness that is explained by the fact why the prosodic model does not take yet counts the micro-prosodic phenomena. The pauses are well located and perceived well. The objective was to conceive and carry out a system of voice synthesis starting from diacritized Arab texts as understandable and naturalness as possible. The first criterion of intelligibility is reached by to the linguistic module of treatment, which makes it possible to carry out the pre-treatments necessary on the chain of entry to be synthesized and to carry out conversion graphemes to phonemes.

A new technology of VoiceXML as a markup language has presented among human speech over the network. Some suggestions to add Arabic support to this technology [18]. They introduce this technology and its methodology (how to work). Since that this technology requires the user to own some major

components such as VoiceXML browser, Arabic TTS and Arabic ASR, the paper describes these components in some details. It firstly introduces the VoiceXML, then it explains the architecture of the VoiceXML browser in details, after that it discusses the specifications of Arabic language, next it discusses the method of building Arabic TTS and Arabic ASR and their architecture.
A rule-based hybrid synthesis Arabic TTS system was developed in [19]. Phonemes were the essential elements of the synthesizer, the proposed Arabic TTS system is vocabulary independent with intelligible output speech, so it can handle all types of input text. It has the flexibility of changing the speaker from male to female and other sound variants like whispering. The standard Arabic text is mostly unvowelized; hence the need of vowelization (Tashkeel), the proposed system omits the need to some of the vowelization symbols like sukoon and has the ability to enrich the exception dictionary by listing the exact pronunciation of the common words, there, no need to vowelize them. Comparing with other available Arabic TTS systems, the proposed Hybrid TTS has small size, high accuracy, and vocabulary independence features which make it in general more reliable than other TTS systems. The system is free for distribution and for development.
The objectives in the work of [20] consisted in improving the naturalness of an Arabic Text To Speech Synthesis system (ARAVOICE). To reach this result, a proposed approach for the automatic generation of the stress presented. The rules used in the phonological module are founded primarily on an algorithm of stressing. Its represents the core of the tonal rules which are employed in the phonetic module. The proposed model adapted, in another point, diagrams generated for the text processing and that while acting on the size of the sentences of the text to reading.
In the paper of [21], a proposed mini-transliteration system for Arabic-numeral expressions is presented. It can efficiently and correctly convert Arabic numeral expressions found in Korean text into phonemes for embedded TTS systems. For the purpose of building grapheme-to-phoneme rules, components of ANEs are deduced, and investigated their pattern and arithmetic features based on the analyzed corpus. A word sense disambiguation was developed to resolve ambiguities and minimized the amount of memory used. It reduced the process time without any serious loss of accuracy, and showed high accuracy, [21].

6 CONCLUSIONS

The project analyzed, designed, implemented and tested an Arabic Text-to-Speech (ArTTS) model to be used in Arabic text pronunciation. Introduction to speech is introduced, includs speech recognition and speech synthesizer. The analysis and design models are described, taken in our consideration the object oriented analysis and design, using UML diagrams. Also, the layout structure of ArTTS is designed and implemented using C sharp language in visual studio .Net 2008.
Arabic digits and numbers pronunciation included in ArTTS are associated with related rules. Also, the different rules of Arabic graphemes pronounced letters are included and designed in ArTTS.
ArTTS concatenates these pronounced graphemes and produced sounded Arabic words. The result that produced from numbers matches Arabic pronouncing of numbers.
The result produced from letters is accepted and understood by the listener. ArTTS is dealing with about 90 rules of numbers and diactrize rules. ArTTS can deal with more rules by putting the new rules in (Rule Phase) for both numbers and letters. Using more rules will enhance the output and make it more similar to Arabic pronunciation. Also, ArTTS will be enhanced using prosody technology.

ACKNOLEDGEMENS
I would like to thank IT department, faculty of computing and IT deanship for their support and guidance and insights that contributed tremendously to this paper. Also, I would like to acknowledge Fahd Al-Hiedary, Mansour AlJohany Seef Al-Harthi for their programming efforts.

REFERENCES

[1] Ahmed Mokhtar Omar, (2006). "دراسة الصوت اللغوي". Publisher: Alam Alkotob, Cairo, Egypt.

[2] Mansour Mohammed ALghamdi, (2006). "قوانين الفونولوجيا العربيه", Publisher: King Abdulaziz City for Science and Technology (KACST), Saudi Arabia.

[3] Wikipedia "speech", (15/3/2009): http://en.wikipedia.org/wiki/Speech

[4] Wikipedia "speech technology", (16/3/2009) http://en.wikipedia.org/wiki/Speech_technology

[5] Wisegeek "speech recognition", (17/3/2009) http://www.wisegeek.com/what-is-voice-recognition.htm

[6] Wisegeek "text to speech", (20/3/2009) http://www.wisegeek.com/what-is-tts.htm

[7] Wikipedia "Phoneme", (29/3/2009) http://en.wikipedia.org/wiki/Phoneme

[8] Wikipedia " Allophone", (30/3/2009) http://en.wikipedia.org/wiki/Allophone

[9] Wikipedia " Morpheme ", (19/3/2009) http://en.wikipedia.org/wiki/Morpheme

[10] Wikipedia " syllable", (19/3/2009) http://en.wikipedia.org/wiki/Syllable

[11] Acapela TTS: http://www.acapela-group.com/text-to-speech-interactive-demo.html

[12] Sakhr TTS : http://www.sakhr.com/TTS/TTS.asp

[13] Site Pal TTS: http://www.oddcast.com/home /demos/tts/tts_example.php?

[14] Allan Ramsay and Hanady Mansour (2008). Towards including prosody in a text-to-speech system for modern standard Arabic, Computer Speech and Language 22 (2008) 84–103.

[15] Anna Prˇibilova, and Jirˇı´ Prˇibil, (2006). Non-linear frequency scale mapping for voice conversion in text-to-speech system with cepstral description, Speech Communication 48 (2006) 1691–1703.

[16] Harald Romsdorfer and Beat Pfister, (2007). Text analysis and language identification for polyglot text-to-speech synthesis, Speech Communication 49 (2007) 697–724.

[17] Zemirli, Z. (2006). ARAB_TTS: An Arabic Text To Speech Synthesis. Computer Systems and Applications, 2006. IEEE International Conference.

[18] Kosayba, B.; Alkhedr, H.; Jdeed, F.; Shriedi, A.; Al-mozaien, E. (2008). Arabic Phonetic Web Sites Platform Using VoiceXML, Information and Communication Technologies: From Theory to Applications, 2008. ICTTA 2008. 3rd International Conference.

[19] Mustafa Zeki, Othman O. Khalifa, A. W. Naji, (2010). Development of An Arabic Text-To-Speech System, International Conference on Computer and Communication Engineering (ICCCE 2010), 11-13 May 2010, Kuala Lumpur, Malaysia, IEEE.

[20] Zouhir ZEMIRLI, Salima KHABET, M'hamed MOSTEGHANEM, (2007). An effective model of stressing in an Arabic Text To Speech System. IEEE

[21] Youngim Jung, Aesun Yoon, and Hyuk-Chul Kwon, (2007). Grapheme-to-Phoneme Conversion of Arabic Numeral Expressions for Embedded TTS Systems. IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING, VOL. 15, NO. 1, JANUARY 2007.

Ontology and its Methodology

Susan Fisal Ellakwah [*1], Passent El-Kafrawy[**2], Mohamed Amin[**3], El-Sayed El-Azhary[*4]

[*]*Central Lab for Agricultural Expert Systems (CLAES)*

Agricultural Research Center (ARC)

Giza, Egypt

[1]fisalsusan@yahoo.com

[4]sayed@claes.sci.eg

[**]*Mathematics and CS Department, Faculty of Science,*

Menoufia University, Egypt

[2]passentmk@gmail.com

***Abstract*— Ontology is used for communication between people and organizations by providing a common terminology over a domain. This work presents a method of establishing ontology from existing ontologies. Establishing ontology from scratch is hard and expensive. This work establishes ontology by matching and merging existing ontologies. Ontologies can be matched and merged to produce a single integrated ontology. Integrated ontology has consistent and coherent information rather than using multiple ontologies, which may be heterogeneous and inconsistent. Heterogeneity between different ontologies in the same domain is the primary obstacle for interoperation between systems. Heterogeneity leads to the absence of a standard terminology for any given domain that may cause problems when an agent, service, or application uses information from two different ontologies. Integrating ontologies is a very important process to enable applications, agents and services to communicate and understand each other.**

1 INTRODUCTION

The term ontology refers to a wide range of formal representations, including taxonomies, hierarchical terminology vocabularies or detailed logical theories describing a domain [1]. For this reason, a precise definition of the term is rather difficult whereas different definitions have appeared in the literature. One commonly used definition is based on the original use of the term in philosophy, where ontology is a systematic account of Existence. For artificial intelligence (AI) systems, what “exists” is that what can be represented [2]; therefore, an ontology in the AI context is a structure that specifies a conceptualization, or, more accurately, a specification of a shared conceptualization of a domain [3]. "An Ontology is a formal, explicit specification of a shared conceptualization [4]. *Conceptualization* refers to an abstract model of some phenomenon in the world by having identified the relevant concepts of that phenomenon. *Explicit* means that the type of concepts used, and the constraints on their use, are explicitly defined. *Formal* refers to the fact that the ontology should be machine-readable. *Shared* reflects the notion that an ontology captures consensual knowledge, that is, it is not private of some individual, but accepted by a group.

In recent years, the AI community has borrowed that term and used it to refer to a set of concepts or terms that are useful in describing or modeling a certain area of knowledge. It provides a source of precisely defined terms and relations. The role of ontologies is to capture domain knowledge in a generic way, provide a commonly agreed upon understanding of a domain and transform the implicit into the explicit knowledge.

Two important challenges facing current communities of researchers and practitioners in the field of software engineering and technology (SET) are knowledge integration and computer-based automatic support. The first challenge implies wasting a lot of time and effort and this is due to one of the difficulties in human relationships, namely the lack of explicit knowledge shared among members of a group/project, with other groups and with other stakeholders. The second challenge arises as many projects include the design and/or construction of advanced tools for supporting different software engineering activities. These tools should provide as much functionality as possible with the smallest cost of development. Both challenges can be solved and more easily approached by using ontologies.

Ontologies provide a shared and common understanding of a domain that can be communicated between people and across application systems. Ontologies play a major role in supporting information exchange processes in various areas. Informally, ontology specifies common vocabulary between different systems. It tries to identify and overcome the barriers for sharing and reusing knowledge represented by AI programs. Ontology is used for knowledge sharing and reuse. It improves information organization, management and understanding.

The starting point for creating ontology could arise from different situations. An ontology can be created from scratch; from existing ontologies (whether global or local ontologies); from a corpus of information sources; or a combination of the latter two approaches. Various degrees of automation could be used to build ontology, ranging from fully manual, semi-automated, to fully-automated. At present, the fully automated method only functions well for very light weight ontologies under very limited circumstances. This work presents semi-automated method for establishing ontology.

Currently, ontologies are widely used in knowledge engineering, artificial intelligence and computer science, in applications related to knowledge management, natural language processing, e-commerce, intelligent integration information, information retrieval, database design and integration, bio-informatics, education, etc.

There are several reasons for developing ontology. First of all, sharing common understanding of the structure of information among people or software agents. Second, enabling the reuse of knowledge. Third, making domain assumptions explicit. Fourth, separating domain knowledge from the operational knowledge. Fifth, analyzing domain knowledge. Sixth, increasing interoperability among various domain of knowledge. Seventh, enhancing scalability of new knowledge into the existing domain. Finally, searching and reasoning a specific knowledge in domain knowledge can be done using ontology.

This section introduces the notion of ontology and the benefits of developing ontologies. Next section defines the main two processes for merging an ontology from preexisting ontologies. In section 3 the related work is reviewed. Section 4 presents the proposed semi-automated technique for developing ontology. Finally, in section 5 the conclusion and future work.

2 ONTOLOGY MATCHING AND MERGING

Ontologies, which are used in order to support interoperability and common understanding between the different parties, are a key component in solving the problem of semantic heterogeneity, thus enabling semantic interoperability between different web applications and services. Recently, ontologies have become a popular research topic in many areas, including electronic commerce, knowledge management, knowledge engineering and natural language processing. Ontologies provide a common understanding of a domain that can be communicated between people, and of heterogeneous and widely spread application systems. In fact, they have been developed in Artificial Intelligence (AI) research communities to facilitate knowledge sharing and reuse. The goal of an ontology is to achieve a common and shared knowledge that can be transmitted between people and between application systems. Thus, ontologies play an important role in achieving interoperability across organizations and on the Semantic Web because they aim to capture domain knowledge and their role to create semantics explicitly in a generic way and provide the basis for agreement within a domain. Ontology is used to enable interoperation between Web applications from different areas or from different views on one area. For that reason, it is necessary to establish mappings among concepts of different ontologies to capture the semantic correspondence between them. However, establishing such a correspondence is not an easy task.

Multiple ontologies need to be accessed from different systems; the distributed nature of ontology development has led to dissimilar ontologies for the same or overlapping domains. Thus, various parties with different ontologies do not fully understand each other. To solve these problems, it is necessary to use ontology matching and geared for interoperability. Ontology matching aims at finding correspondences between entities of different ontologies, these correspondences may stand for equivalence as well as other relations between ontology entities. Matching is an essential aspect of merging and could also be used to initiate merging. Ontology merging is a first natural use of ontology matching. Ontology merging is the process of creating a new single coherent ontology from two or more existing source ontologies related to the same domain, the new ontology replaces the source ontologies. This paper presents a system to establish global ontology in specific domain from existing ontologies by matching and merging techniques to obtain a high quality result. Global ontology allows users to avoid querying the local ontologies one by one, and to obtain a result from them just by querying a global ontology. Global ontology has standard and shared terminology. It is consistent and coherent. It has no redundancy. This section presents some of the basic matching methods for assessing the similarity or the relations between ontology entities. These basic techniques that can be used for building correspondences based on terminological, conceptual, extensional and semantic arguments. The following basic techniques are the building blocks on which a matching solution is built, these techniques cannot be used in isolation, but that each of them can take advantage of the results provided by the others. Another part of the art of ontology matching relies on selecting and combining these methods (matchers).

a. Name-based techniques

Name-based techniques compare strings. They can be applied to the name, the label or the comments of entities in order to find those which are similar. This can be used for comparing class names and/or URIs.

- String-based methods

String-based methods are often used in order to match names and name descriptions of ontology entities. These techniques consider strings as sequences of letters in an alphabet. They are typically based on the following intuition: the more similar the strings, the more likely they are to denote the same concepts. Usually, distance functions map a pair of strings to a real number, where a smaller value of the real number indicates a greater similarity between the strings. Some examples of string-based methods which are extensively used in matching systems are prefix, suffix, edit, and n-gram distances.

- Language-based methods

Language-based methods rely on using natural language processing techniques to help extract the meaningful terms from a text. Terms are phrases that identify concepts; they are often used for labeling concepts in ontologies. Comparing these terms and their relations should help to assess the similarity of the ontology entities they name and comment upon. These are based on linguistic knowledge; indeed, there are two general techniques, one relying on algorithms only, while the other can use external lexicon-based resources such as dictionaries. As a consequence, ontology matching can take great advantage of recognizing and identifying them in strings; in other words, it uses the internal linguistic properties of the instances, such as their syntactic properties. In general, extrinsic linguistic methods are used for finding extra similarities between terms; external linguistic resources such as dictionaries and lexicons increase the chances of finding matching terms. Lexical information (e.g. names, definitions and distance between strings) can help with reclassification of matching results. Auxiliary information (e.g. WorldNet) provides semantics for the elements in ontologies.

b. Structure-based techniques

In this method of matching, instead of comparing their names or identifiers, the structures of entities that can be found in an ontology are compared. In this kind of matcher, information is used about the structure, such as subclass and super-class relationships, domain and range of properties, and the graph structure of ontologies. In fact, this information provides insight into ontologies. This kind of comparison can be divided into the following:

- Internal Structure: this method is comparing the internal structure of entities, in other the words, the similarities between the names of their properties (e.g., the value range or cardinality of their attributes).
- External Structure: this method is comparing the relations of the classes with other classes like compute the similarity the super-classes of the two classes.

c. Extensional techniques

In a case where two ontologies have similar instances, finding corresponding concepts based on checking similarities between the individuals is required. If the similarity level of two instances reaches a threshold, then the two individuals can be considered as matched. The matching can be based on instance comparisons. To identify the similarity level of two instances, string similarity methods may be used.

d. Semantic-based techniques

The key characteristics of semantic methods are that model-theoretic semantics is used to justify their results. They are deductive methods. Of course, pure deductive methods do not perform very well alone for an essentially inductive task like ontology matching. They hence need a preprocessing phase which provides ‘anchors’, i.e., entities which are declared, for example, to be equivalent (based on the identity of their names or user input for instance).The semantic methods act as amplifiers of these seeding alignments.

3 RELATED WORK

Several tools exists for ontology establishment, ranging from fully manual to fully automated. Many of the semi-automated ontology merging and alignment tools are listed in this section. PROMPT [5] begins with the linguistic-similarity matches for the initial comparison, but generates a list of suggestions for the user based on linguistic and structural knowledge and then points the user to possible effects of these changes. SMART [22] looks for linguistically similar class names through class-name matches, creates a list of initial linguistic similarity (synonym, shared substring, common suffix, and common prefix) based on class-name similarity, studies the structures of relation in merged concepts, and matches slot names and slot value types. It makes suggestions for users, checks for conflicts, and provides solutions to these conflicts.

OntoMorph [6] provides a powerful rule language for specifying mappings, and facilitates ontology merging and the rapid generation of knowledge-base translators. It combines two powerful mechanisms for knowledge-base transformations such as syntactic rewriting and semantic rewriting. Syntactic rewriting is done through pattern-directed rewrite rules for sentence-level transformation based on pattern matching. Semantic rewriting is done through semantic models and logical inference. A concept hierarchy management for ontology alignment and merging is provided in Hierarchical Concept Alignment system [7] (HICAL), where one concept hierarchy is aligned with another concept in another concept hierarchy. HICAL uses a machine-learning method for aligning multiple concept hierarchies, and exploits the data instances in the overlap between the two taxonomies to infer mappings. It uses hierarchies for categorization and syntactical information, not similarity between words, so that it is capable of categorizing different words under the same concept.

Another system that employs machine learning techniques to find ontology mappings is GLUE [8]. If given two ontologies, for each concept in one of the ontologies, GLUE finds the most similar concept in the other one. GLUE works with several similarity measures that are defined with probabilistic definitions. Multiple learning strategies exploit different types of information from instances or taxonomy structures. GLUE can also use common sense knowledge and domain constraints instead of relaxation labeling. It is a well-known constraint optimization technique adapted to work efficiently. Quick Ontology Mapping (QOM)[9] is based on the hypothesis that mapping algorithms can be streamlined so that the loss of quality is marginal, but the improvement of efficiency is tremendous for the ad-hoc mapping of large-size light-weight ontologies. The process model shown in Figure 1 defines QOM in steps.

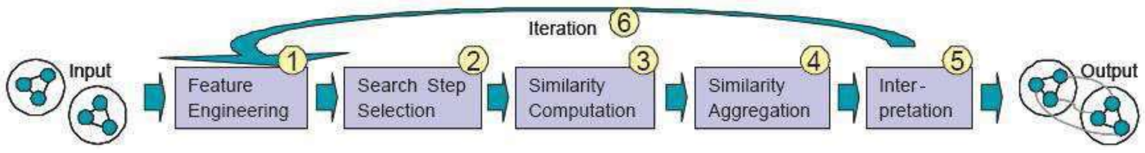


Figure 1: Quick ontology mapping

A generic ontology mapping system, called LILY [10], is based on the extraction of semantic subgraph. LILY exploits both linguistic and structural information in semantic subgraphs to generate initial alignments. After that, a subsequent similarity propagation strategy is applied to produce more alignments if necessary. Finally, LILY uses the classic image threshold selection algorithm to automatically select the threshold, and then extracts final results based on the stable marriage strategy. LILY has different functions for different kinds of tasks: for example, Generic Ontology Matching method (GOM) is used for common matching tasks with small size ontologies; Large scale Ontology Matching method (LOM) is used for matching tasks with large size ontologies; and Semantic Ontology Matching method (SOM) is used for discovering the semantic relations between ontologies. The two limitations of LILY are that it requests the user to manually set the size of subgraph according to different mapping tasks and the efficiency of semantic subgraph is very low in large-scale ontologies.
A multi-strategy learning approach is employed in Learning Source Description (LSD) [12]. In the training phase, LSD chooses one learner among several base learners, such as Name Learner, Content Learner, and XML Learner, based on their confidence score. In the matching phase, ontology matching is performed using the learner chosen in the training phase, in order to improve matching accuracy.

Mediator Environment for Multiple Information Source (MOMIS) [13] creates a lexical matrix using WordNet. A lexical matrix consists of rows storing word forms and columns storing word meanings. Relations between words such as SYN (synonyms), BT (broader terms), NT (narrower terms), and RT (related terms) can be represented using the developed lexical matrix.
LOM (Lexicon-based Ontology Mapping) [14] LOM is a semi-automatic lexicon-based ontology-mapping tool that supports a human mapping engineer with a first-cut comparison of ontological terms between the ontologies to be mapped, based on their lexical similarity. It proposes four phases of a lexical similarity measurement in ontology matching: (1) Whole term matching, (2) Word constituent matching, (3) Synset matching, and (4) Type matching.

Combining match algorithms (COMA++) [15] is a well-known ontology-matching tool providing a graphical user interface. COMA++ adopts multi-match strategies to perform matching of relational schemas, W3C XSD, and OWL. The implemented multi-match strategies utilize fragment-based matching and reuse-oriented matching. Unlike the previous ontology matching methods focusing on 1:1 mapping. Risk Minimization based Ontology Mapping (RiMOM) [16] automates the process of discoveries on 1:1, n:1, 1:null and null:1 mappings. Besides, RiMOM solves the problem of name conflicts in mapping process using thesaurus and statistical techniques. Integrated Learning In Alignment of Data and Schema (ILIADS) [17] proposes an ontology matching algorithm based on logical inference. ILIADS integrates the previous ontology matching approach and the logical inference similarity measure to achieve better matching of ontologies. ONtology compositION system (ONION) [18] resolves terminological heterogeneity in ontologies and produces articulation rules for mappings. The linguistic matcher identifies all possible pairs of terms in ontologies and assigns a similarity score to each pair. If the similarity score is above the threshold, then the match is accepted and an articulation rule is generated. After the matches generated by a linguistic matcher are available, a structure-based matcher looks for further matches. An inference-based matcher generates matches based on rules available with ontologies or any seed rules provided by experts.

Multiple iterations are required for generating semantic matches between ontologies. A human expert chooses, deletes, or modifies suggested matches using a GUI tool. A linguistic matcher fails when semantics should be considered. CROSI Mapping System (CMS) [19] is an ontology alignment system. It is a structure matching system on the rich semantics of the OWL constructs. Its modular architecture allows the system to consult external linguistic resources and consists of feature generation, feature selection, multi-strategy similarity aggregator, and similarity evaluator. FCA-Merge [20] is a method for ontology merging based on Ganter and Wille's formal concept analysis-28, lattice exploration, and instances of ontologies to be merged.

The overall process of ontology merging consists of three steps: 1) instance extraction and generation of the formal context for each ontology, 2) the computation of the pruned concept lattice by algorithm TITANIC29, and 3) the non automatic generation

of the merged ontology with human interaction based on the concept lattice. CHIMAERA [21] is an interactive ontology merging tool based on the *Ontolingual* ontology editor. It makes users affect merging process at any point during merge process, analyzes ontologies to be merged, and if linguistic matches are found, the merge is processed automatically, otherwise, further action can be made by the use. It uses subclass and super-class relationship. Ontology Mapping Enhancer (OMEN) [23] is a probabilistic ontology mapping tool which enhances the quality of existing ontology mappings using a Bayesian Net. The Bayesian Net uses a set of meta-rules that represents how much each ontology mapping affects other related mappings based on ontology structure and the semantics of ontology relations. Existing mappings between two concepts can be used for inferring other mappings between related concepts. P2P ontology mapping [24] proposes a framework that allows agents to interact with other agents efficiently based
on the dynamic mapping of only the portion of ontologies relevant to the interaction. The framework executes three steps: 1) Generates the hypotheses; 2) Filters the hypotheses; and 3) Selects the best hypothesis.

4 METHODOLOGY OF ESTABLISHING ONTOLOGY

This section presents a new semi-automated method for establishing global ontology by merging pre-existing ontologies. This technique consists of two main components: matching process and merging process.

A. *System Structure*

The structure of the two main components, matching process and merging process, are shown in figure 2. Ontology matching tries to identify similarities between heterogeneous ontologies and to automatically create suitable mappings for transformation. Matching is an essential aspect of merging and could also be used to initiate merging. Ontology merging is the process that will create a single global coherent ontology by unifying two or more existing ontologies.

B. *System Components*

As mentioned before the system is composed of two main components: the matching process component and the merging component.

1) *Matching Process:* Matching process component receives source ontologies and computes similarities between the ontological entities. A matching process uses an algorithm or matchers. This allows selection of the matchers depending on the application domain and schema types. The ontological entities in an ontology consists of concepts, properties and values [25]. Matching process consists of two parts: The first part (I) computes alignment suggestions and the second part (II) interacts with the user to decide on the final alignments as shown in figure 3.

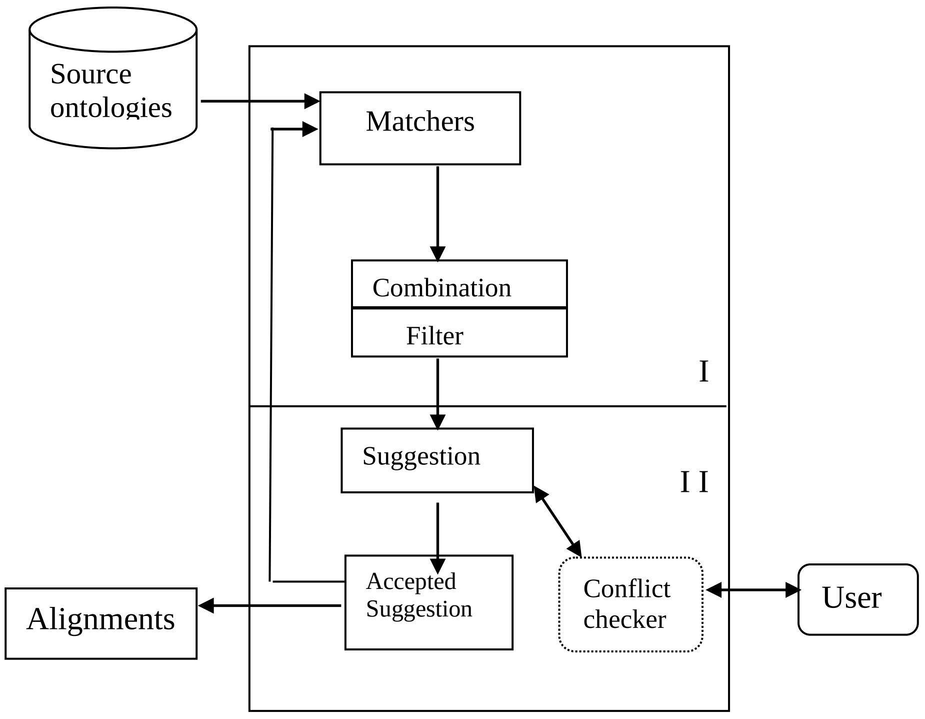


Figure 3 matching process components

An alignment algorithm includes several matchers. 'Matchers' calculate similarities between the ontological entities from the different source ontologies. The matchers implement the strategies based on linguistic or structure techniques. 'Combination' combines matchers' similarities and derive the correspondences between the entities. 'Filter' filters out the matched pairs. 'Suggestions' are determined by combining and filtering the results generated by one or more matchers. The interactive component of the alignment algorithm presents the suggestions to the user who accepts or rejects them. The acceptance and rejection of a suggestion may influence further suggestions. Conflict checker is used to avoid conflicts introduced by the alignment relationships. The output of the alignment algorithm is a set of alignment relationships between terms from the source ontologies. Alignments are a set of correspondences between two or more ontologies. The 'alignments' is the output of the matching process.

2) *Merging Process:* a global merged ontology is to be established from the source ontologies and their identified alignments, as shown in figure 4. 'Checker' is used to avoid conflicts as well as to detect non-satisfying concepts and, if so desired by the user, to remove redundancies.

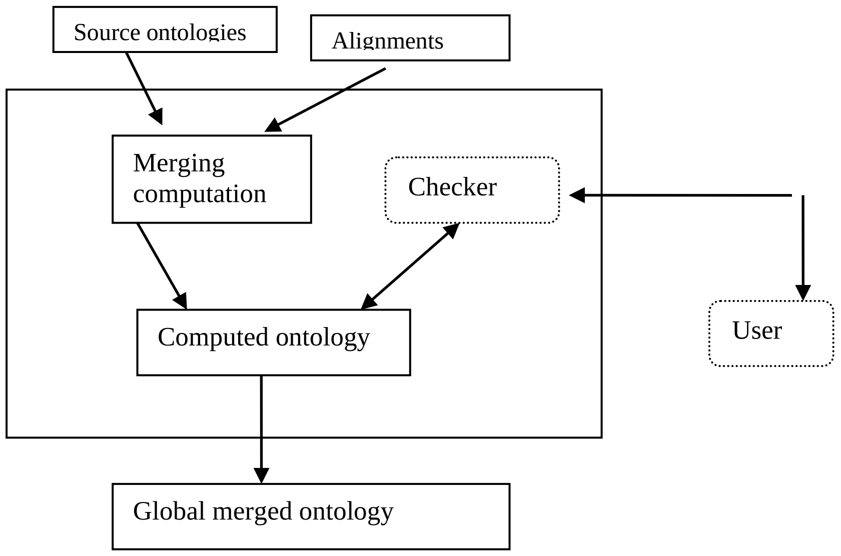


Figure 4 merging process components

5 CONCLUSIONS

This paper presents a system to build a global ontology from different ontologies in the same domain**.** The process of building ontology is a high-cost process. Building global ontology from scratch is hard, cost and time-consuming. This work presents a method for reusing and sharing existing ontologies by matching and merging them.

In the future work, we will apply this method on the agricultural domain to obtain a global agricultural domain. The algorithms of matching and merging will be established and implemented. The agricultural domain will be established by using not only existing ontologies but also we will establish a tool to help in building the not existing ontologies and integrating them together.

ACKNOWLEDGMENT

I would like to thank Dr. Passent El-Kafrawy and Dr. Mohamed Amin. Special thanks to Dr. El-Sayed El-Azhary. Many thanks to my friends at CLAES.

REFERENCES

[1] Noy, N. & Klein, M. "Ontology Evolution: Not the Same as Schema Evolution" Knowledge and Information Systems, 6 (4), pp. 428-440, 2004. also available as SMI technical report SMI-2002-0926.
[2] Gruber, T.R. "A Translation Approach to Portable Ontology Specifications" Knowledge Acquisition, 5 (2), pp.199-220, 1993.
[3] Gruber, T.R. "Toward Principles for the Design of Ontologies Used for Knowledge Sharing," Formal Ontology in Conceptual Analysis and Knowledge Representation" 1993, also available as Technical Report KSL-93-04, Knowledge Systems Laboratory, Stanford University
[4] Borst P, Akkermans H, Top J. "Engineering ontologies," International Journal of Human-Computer Studies 46:pp.365–406,1997.
[5] N. Noy and M. Musen, "PROMPT: Algorithm and Tool for Automated Ontology Merging and Align,ment," in Proc.of17th National Conference on Artificial Intelligence,(AAAI,), pp. 450–455, Austin, Texas 2000.

[6] H. Chalupsky. "Ontomorph: A Translation System for Symbolic Knowledge", Principles of Knowledge Representation and Reasoning, 2000.

[7] R. Ichise, H. Takeda, and S. Honiden. "Rule Induction for Concept Hierarchy Alignment", Proceedings of the Workshop on Ontology Learning at the 17th International Joint Conference on Artificial Intelligence (IJCAI), 2001.

[8] An-Hai Doan, J. Madhavan, Pedro Domingos, and Alon Halevy. "Learning to map ontologies on the semantic web,". In Proceedings of the International World Wide Web Conference (WWW), pages 662–673, 2003.

[9] Marc Ehrig and Steffen Staab. "QOM: Quick ontology mapping,". In Proceedings of the 3rd International Semantic Web Conference (ISWC), pages 683–697, 2004.

[10] Peng Wang and Baowen Xu."Lily: Ontology alignment results for oaei 2009.In Shvaiko et al. [11].

[11] Pavel Shvaiko, J´erˆome Euzenat, Fausto Giunchiglia, Heiner Stuckenschmidt, Natalya Fridman Noy, and Arnon Rosenthal, editors. Proceedings of the 4th International Workshop on Ontology Matching (OM-2009) collocated with the 8th International Semantic Web Conference (ISWC-2009) Chantilly,USA, October 25, 2009, volume 551 of CEUR Workshop Proceedings.CEUR-WS.org, 2009.

[12] AnHai Doan, Pedro Domingos, and Alon Halevy, "Learning to Match the Schemas of Data Sources: A Multistrategy Approach," Machine Learning, Vol. 50, No.3, pp 279-301(2003)

[13] Domenico Beneventano, Sonia Bergamaschi, Francesco Guerra, and Maurizio, (2003) "Synthesizing an Integrated Ontology,"IEEE Internet Computing, pp 42-51.

[14] John Li, "LOM: A Lexicon-based Ontology Mapping Tool," Proc. of the Performance Metrics for Intelligent Systems Workshop (PerMIS. '04), pp 1-5, 2004. and Erhard Rahm, (2005) "Schema and ontology matching

[15] David Aumueller, Hong-Hai Do, Sabine Massmann, and Erhard Rahm, "Schema and ontology matching with COMA++ ,"Proc. of the ACM SIGMOD International Conference on Management of Data, pp 906-908, 2005

[16] Jie Tang, Juan-Zi Li, Bangyong Liang, Xiaotong Huang, Yi Li and Kehong Wang,"Using Bayesian decision for ontology mapping," *Journal of Web Semantics*, Vol. 4, No. 4, pp 243-262 (2006).

[17] Octavian Udrea, Lise Getoor, and Renée J. Miller, "Leveraging data and structure in ontology integration," Proc. Of ACMSIGMOD International Conference on Management of Data, pp 449-460, 2007.

[18] Mitra, P and Wiederhold, G, "Resolving Terminological Heterogeneity in Ontologies", Proceedings of the ECAI'02 workshop on Ontologies and Semantic Interoperability, 2002.

[19] Yannis Kalfoglou, Bo Hu, "CROSI Mapping System(CMS) Results of the 2005 Ontology Alignment Contest",K-CAP Integrating Ontologies Workshop 2005, Banff, Alberta, Canada, 2005.

[20] Gerd Stumme, Alexander Maedche, "FCA-Merge: Bottom-Up Merging of Ontologies," In proceeding of the International Joint Conference on Artificial Intelligence IJCA101, Seattle, USA, 2001.

[21] D. McGuinness, R. Fikes, J. Rice, and S.Wilder,"The Chimaera Ontology Environment", In Proceedings of the 17th National Conferenceon Artificial Intelligence (AAAI), 2000.

[22] Natalya Fridman Noy and Mark A. Musen,"Smart:Automated Support for Oontology Merging andAlignment", Proceedings of the Twelfth Banff Workshopon Knowledge Acquistion, Modeling, and Management,Banff Algeberta, 1999.

[23] Prasenjit Mitra, Natasha F. Noy, Anju Jaiswals, "OMEN: A Probabilistic Ontology Mapping Tool", International Semantic Web Conference 2005: 537-547.

[24] 32. Paolo Besana, Dave Robertson, and MichaelRovatsos, *"Exploiting interaction contexts in P2P ontology mapping"*, P2PKM 2005.

[25] Bon J. Wielinga, Expertise Model Definition Document, University of Amsterdam, 1994.

Improving YamCha Tool Performance

Salwa Hamada

Electronics Research Institute (ERl)

hesalwa@hotmail.com

***Abstract*---NE involves identification of proper names in texts, and classification into a set of predefined categories of interest. Three universally accepted categories: person, location and organization. Other common tasks: recognition of date/time expressions, measures (percent, money, weight etc), email addresses etc. YamCha is a Named Entity Recognition (NER) tool available as an open source to help the researchers [1]. Unfortunately the YamCha tool cannot deal with words that are preceded by Arabic prepositions or conjunctions if there is no space between the word and any of them [2]. This is because YamCha's model cannot detect that the preposition or conjunction character is not a part of the word. This paper presents a hypothesis that improves the YamCha's tool performance. It assumed that we can improve the performance of YamCha by using the RDI-POST in the preprocessing phase. It claims that it is possible to raise the accuracy rate of YamCha by applying some preprocessing steps to YamCha's input text. The paper also presents an experiment that shows that by applying this preprocessing step, the accuracy rate of the YamCha tool increased from 57% to 88%.**

1 INTRODUCTION

The YamCha tool is based on making a training phase and generating a model to test the corpus. It is created as generic, customizable, open source text chunker. It can be adapted to a lot of other tag-oriented NLP tasks. It uses state-of-the-art machine learning algorithm called Support Vector Machines (SVMs), first introduced by Vapnik in 1995[3]. YamCha tool takes the model and data in Roman format as inputs. In this paper Yassine Benajiba model had been used for training data. The RDI Part-Of-Speech (POS) tagger RPOST tool has been used to tag parts of speech before using YamCha tool. Then some devolved program had been applied to YamCha's input text after that. To use YamCha first we use the RPOST. Second a converter program to convert from Arabic to Roman format. YamCha generates it output in Roman format, so we used a program to convert Roman characters to Arabic ones so that the output will de readable as shown in section4.
The rest of this paper is divided as follows: section two descibes RPOST, section three presents that proposed steps for improving the YamCha tool. YamCha tool in processing is discussed in section four. Section five shows how to change YamCha output to a user readable format. A case study has been applied in section six and followed by an Evaluation to YamCha performance. Section seven introduces the final conclusion.

2 THE TAGGER YAMCHA TOOL

The NER task is one of the most important subtasks in Information Extraction. It is defined as the identification and classification of Named Entities (NE's) within an open-domain text. YamCha is a generic, customizable, and open source text chunker-oriented towards a lot of NLP tasks, such as POS tagging, Named Entity Recognition, base NP chunking, and Text Chunking. YamCha uses SVM a state-of-the-art machine learning algorithm.

In this paper Yamcha is used to tag named entities. As we mention before YamCha cannot deal with word preceded with a preposition or a conjunction [2].

for example:

مصر والاردن وايران والصين

YamCha succeeded to tag the first word "مصر" as a location but it fails to tag the rest of locations "الاردن" , "ايران" and "الصين" because it considers the preceding "و" as a part of the word". RPOST can help in that as it can define the conjunctions and the prepositions as separate characters.

This paper focuses on improving YamCha's performance with conjunctions and preposition characters. Let us take the و /AlwAw/ ('and') as a case example. The /AlwAw/ ('and') is the most commonly used coordinating conjunction in Arabic and a common source of morphological ambiguity. According to a manual evaluation of a random sample of 100k Arabic tokens from newswire articles [4], it has been found that /AlwAw/ ('and') alone accounts for approximately 8.6% of any written text (i.e. every 100 words include و and 8.6 /wAw/ ('and')).

3 RDI POS TAGGER TOOL (RDI-POST) [5]

RDI's NLP core engine can carry out Arabic morphological analysis, Arabic POS tagging, and Arabic Lexical Semantic Analysis. Arabic POS tagging is a fundamental linguistic analysis process where POS tags that convey the basic context-free syntactic features of input surface text words are extracted. POS tags are the most essential input features for all kinds of natural language computational syntax parsers which are a step towards language understanding and machine translation as well. Based on that definition the position of POS tagging is obviously a middle sub layer between the two fundamental lexical and syntactic ones on the NLP ladder

Appendix (1) below shows a sample of Arabic POS tags set along with the meaning of each tag verbalized in both English and Arabic. The RDI-POST tagger tool tags each word with its appropriate features tag as shown in the below example.

Example:
The input sentence is "إحالة الجنود"
The output:
إحالة{ NullPrefix Noun NounInfinit Femin Single }الجنود{Definit Noun NullSuffix}

4 STEPS TO IMPROVE YAMCHA PERFORMANCE:

A. Collecting the data: Steps has been mentioned in details in [1].

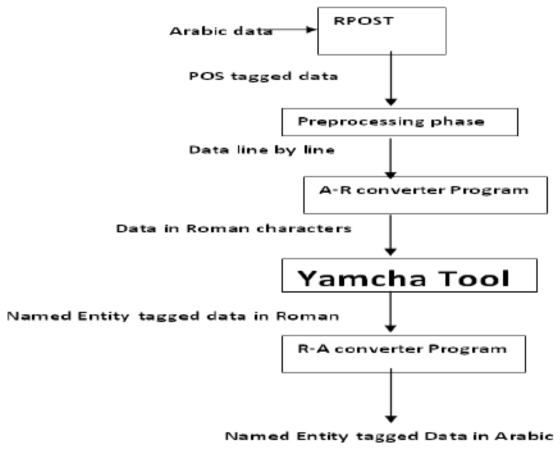
B. YamCha tool features:

1) Moderately high performance chunker based on Support Vector Machines (SVM).
2) Independent from the given task, training/testing with any data which can be seen as a "generic" text chunking task.
3) Use PKE/ PKI, which make the classification (chunking) speed faster than the original SVMs.
4) Can redefine feature sets (window-size), parsing-direction (forward/backward) and algorithms of multi-class problem (pair wise/one vs. rest).
5) Practical chunking time (1 or 2 sec. /sentence. it highly depends on the task).
6) Can perform partial chunking.

C. Input

The input of **YamCha** tool is two parameters. The first is the model, this model represent the training data. In this case use Yassine Benajiba Model. The second input is the data file in roman format.

D. The Proposed steps



E. Filtering Data

1) We use a java program to change RPOST output format to YamCha format which need Roman letters

The RPOST output produces in the form:

مصر{ اسم معرف }و { حرف عطف } اليمن{اسم معرف}

2) Then we need to split conjunction and the following word in different lines to be accepted in YamCha tool as shown below:

مصر

و

اليمن

3) We use a java language program to remove the arches and put each word in a different line.

F. Character Converter (Convert from Arabic to roman):

YamCha can only process Roman characters, so we've implemented a java program to convert Arabic characters to roman characters. Appendix (2) presents a table that shows each Arabic character with its corresponding Roman one.

1) Program input file format: (Plain Arabic text)

أفاد مصدر مسؤول في وزارة الداخلية امس بالعثور على 50 جثة مجهولة الهوية في العاصمة بغداد

2) Program output file format:

The Roman transliteration (file output)	The Arabic word
>fAd	أفاد

mSdr	مصدر
ms&wl	مسؤول
fy	في
wzArp	وزارة
AldAxlyp	الداخلية
Ams	امس
bAlEvwr	بالعثور
ElY	على
50	50
Hvp	جثة
mHhwlp	مجهولة
Alhwyp	الهوية
Fy	في
AlEASmp	العاصمة
bgdAd.	بغداد

G. Test file formats [2]:

Generally speaking, training and test files must consist of multiple tokens. In addition, a token consists of multiple (but fixed-numbers) columns. The definition of tokens depends on tasks; however, in most of typical cases, they simply correspond to words. Each token must be represented in one line, with the columns separated by white space (spaces or tabular characters). A sequence of token becomes a sentence. To identify the boundary between sentences, just put an empty line (or just put 'EOS'). However the number of columns must be fixed through all tokens. Furthermore, there are some kinds of "semantics" among the columns. For example, 1st column is ‘word’; second column is 'POS tag' third column is 'sub-category of POS' and so on.

Symbols of the output:

- B-ORG: represents the beginning of an organization’s name
- I-ORG: represents the middle Intermediate of an organization’s name
- B-PER: represents beginning of a person’s name
- I-PER: represents the middle of a person’s name
- B-LOC: represents beginning of a location’s name

5 A CASE STUDY APPLYING PAPER APPROACH YAMCHA TOOL

A. The input:

As mentioned before the input represents each word in a single line and the file must be in roman format (output of the program in section 4.2).

B. The output:

The word	The tagging symbol
>fAd	O
MSdr	O
ms&wl	O
fy	O
wzArp	B-ORG
AldAxlyp	I-ORG
Ams	O
BAlEvwr	O
ElY	O
50	O
Hvp	O
MHhwlp	O
Alhwyp	O
Fy	O
AlEASmp	O
bgdAd.	B-LOC

C. Support Vector Machines (SVM) and Tagging scheme in the IOB format[4]

The NER task in Arabic is relatively different from performing the task in English due to the inherent characteristic linguistic differences of Arabic, most notably, the lack of a direct signal such as capitalization in Arabic orthography to mark a named entity. This tool solves the problem of NER for Arabic. We adopt a discriminative approach to the NER problem. SVMs have yielded the best results when compared to other machine learning approaches to the problem. SVMs are robust to noise in the data and they have powerful generalization ability especially in the presence of a large number of features. Moreover, SVMs have been used successfully in many NLP areas of research in general, and for the NER task in particular YamCha is one of them that converts the NER task to a chunking task using the Inside-Outside-Beginning (IOB) tagging scheme.

Tagging scheme in the IOB format:

Arabic	Roman	English Trans.	Tag
وفي	Wfy	And in	O
بيروت	Byrwt	Beirut	B-LOC
,	,	,	O
وصف	wSf	Described	O
فؤاد	F&Ad	Fouad	B-PER
السنيورة	Alsnywrp	Siniora	I-PER
رئيس	R}ys	president	O
الوزراء	AlwzrA’	The ministers	O
اللبناني	AllbnAny	Lebanese	O

D. Changing YamCha output to a user readable format

Now after the NER task to the input file using YamCha tool, we need to change the output format in a form that the user can read and understand. This is done by what we called a format adaptor program which returns the Arabic character and put the file in XML format.

The Input of program is the output of the program in section 4.4.

The output of program:

<\O>مسؤل <O> <\O> مصدر<O> <\O>افاد <O>

<\O>امس<O> <\ORG>وزارة الداخليه<ORG> <\O>فى <O>

<\O>على<O> <\O>بالعثور<O> <\O>مجهولة<O> <\O>جثه<O> <\O>50<O>

<LOC>بغداد<LOC> <\O>العاصمه<O> <\O>فى<O> <\O>الهويه<O>

6 A CASE STUDY AND EVALUATION

Our hypothesis has been tested using a random sample of 2000 tokens from newswire articles (1998), among which one finds 353 instances of /AlwAw/. This sample has been manually evaluated. Table1 shows the results of using our preprocessing task by comparing output of YamCha before and after applying our preprocessing. Improving tokenization has reduced error rate in POS tagging by approximately 90%. Examining errors in our output, we have found that they are due to a problem related to the training data. May be it does not cover all names.

1) A sample of the output before improving the YamCha tool’s performance:

Words	**Output from** YamCha **before improving**
رحبت	O
مصر	B-LOC
بالتعاون	O

مع	O
سوريا	B-LOC
والاردن	O
ولبنان	O
والعراق	O
لمساعدة	O
الفلسطنين	O
.	O

2) The same sample of the output after improving the YamCha's tool performance:

Words	**Output of** YamCha **after improvement**
رحبت	O
مصر	B-LOC
ب	O
التعاون	O
مع	O
سوريا	B-LOC
و	O
الاردن	**O**
و	O
لبنان	B-LOC
و	O
العراق	B-LOC
ل	O
مساعدة	O
الفلسطنين	O
.	O

7 FINAL RESULTS:

	Precision	**Recall**	**F-measure**
YamCha before our preprocessing	100	40	57
YamCha after our preprocessing	100	80	88

Table1: Experimental Results

8 CONCLUSION:

We believe that incremental development where existing tools are built upon and improved is a much better approach than -inventing tools to address limitations of existing ones. In this paper, we assumed that we can improve the performance of YamCha by using the RDI-POST in the preprocessing phase. This can be applied to all clitics, such as coordinating conjunctions, prepositions; pronouns, etc. Testing and the results it had been found that the preprocessing task succeeded in raising the accuracy rate from 57% to 88%.

ACKNOWLEDGMENT

Thanks a lot to the programmers Gada Mohamed Fathi and Maii Gad who helped in writing the programs code.

REFERENCES

[1] YamCha: Yet Another Multipurpose CHunk Annotator,
http://chasen.org/~taku/software/yamcha/

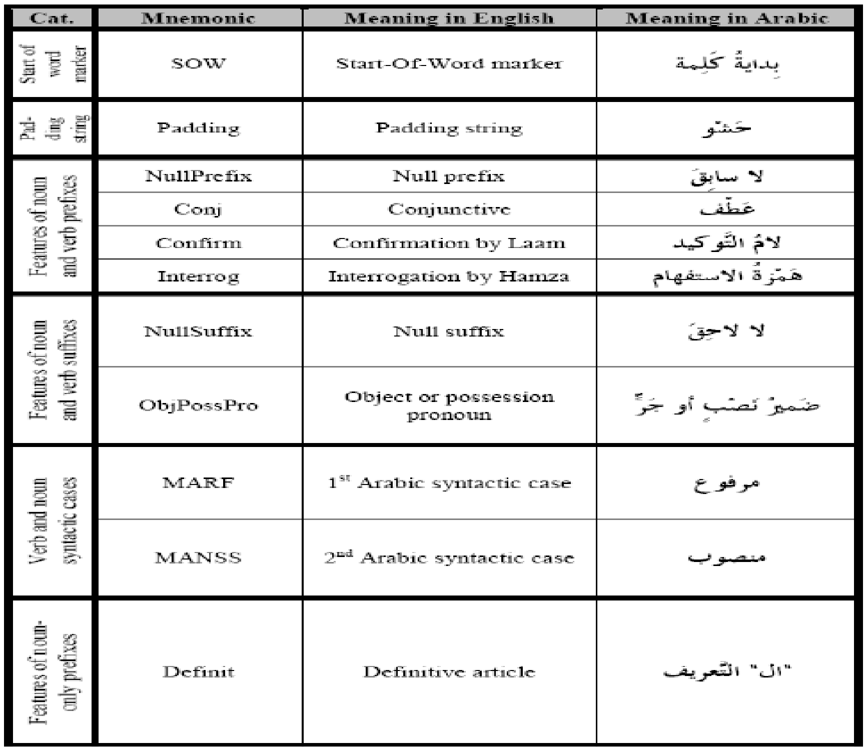
[2] Dina Mostafa Mahmoud Nassif, Ghada Mohamed Fathy Mohamed, Rowan Tarek Ahd El-Hamed, Shimaa Hassan Mohamoud, Zeinab Fayez Ahmed, “Search Engine For Emotions In Arabic”, Graduation project, Cairo university,2010.

[3] Yassine Benajiba, Mona Diab, Paolo Rosso, “ARABIC NAMED ENTITY RECOGNITION: AN SVM-BASED APPROACH”, ,2008

[4] Alahram Newspaper: http://www.ahram.org.eg

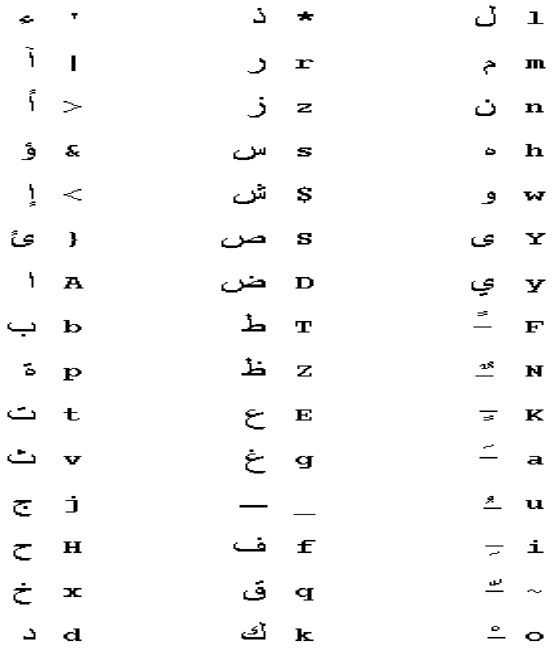
[5] RDI POS tagger tool on:
http://www.rdi-eg.com/technologies/papers.htm ,not available for free.

Appendix(1): A part of RDI Features Table:



Cat.	Mnemonic	Meaning in English	Meaning in Arabic
Start of word marker	SOW	Start-Of-Word marker	بِدايةُ كَلِمة
Pad-ding string	Padding	Padding string	حَشْو
Features of noun and verb prefixes	NullPrefix	Null prefix	لا سابِقَ
	Conj	Conjunctive	عَطْف
	Confirm	Confirmation by Laam	لامُ التَّوكيد
	Interrog	Interrogation by Hamza	هَمْزَةُ الاستفهام
Features of noun and verb suffixes	NullSuffix	Null suffix	لا لاحِقَ
	ObjPossPro	Object or possession pronoun	ضَميرُ نَصْبٍ أو جَرٍّ
Verb and noun syntactic cases	MARF	1st Arabic syntactic case	مرفوع
	MANSS	2nd Arabic syntactic case	منصوب
Features of noun-only prefixes	Definit	Definitive article	"ال" التّعريف

Appendix (2) : Arabic-Roman characters table.



ء	'	ذ	*	ل	l
آ	|	ر	r	م	m
أ	>	ز	z	ن	n
ؤ	&	س	s	ه	h
إ	<	ش	$	و	w
ئ	}	ص	S	ى	Y
ا	A	ض	D	ي	y
ب	b	ط	T	ـً	F
ة	p	ظ	Z	ـٌ	N
ت	t	ع	E	ـٍ	K
ث	v	غ	g	ـَ	a
ج	j	ـ	_	ـُ	u
ح	H	ف	f	ـِ	i
خ	x	ق	q	ـّ	~
د	d	ك	k	ـْ	o

UNL^{+3}: The Gateway to a Fully Operational UNL System

Sameh Alansary[†*1], Magdy Nagi[†**2], Noha Adly[†**3]

[†] *Bibliotheca Alexandrina, P.O. Box 138 El Shatby, Alexandria 21526, Egypt.*

[*] *Department of Phonetics and Linguistics, Faculty of Arts, University of Alexandria*
El Shatby, Alexandria, Egypt
[1]sameh.alansary@bibalex.org

[**]*Computer and Engineering Department, Faculty of Engineering, University of Alexandria, Alexandria, Egypt*
[2]magdy.nagi@bibalex.org
[3]noha.adly@bibalex.org

***Abstract*— In this paper we present the UNL^{+3} program; the latest phase of development in the UNL project. UNL^{+3} has been introduced in order to cure the limitations and problems encountered in the 15-year life of UNL. UNL^{+3} proposes new components to the system in addition to introducing enhancements and improvements to the existing ones. Enhancements encompass the general linguistic infrastructure of the system as well as the specific technicalities of its components. The paper, consequently, compares and contrasts the earlier UNL specifications to the ones proposed by UNL^{+3}, highlighting how they would drastically improve the adequacy and efficiency of the system. More importantly, by putting its propositions into effect, UNL^{+3} would finally render the UNL system fully operational by the end of 2011, ending by that a long phase of experimentation and research, and starting a new phase of development and evolvement. UNL^{+3} is designed and implemented by the UNDL foundation in Geneva, Switzerland and Bibliotheca Alexandrina in Alexandria, Egypt.**

1 INTRODUCTION

The UNL project was launched in 1996 in the Institute of Advanced Studies in the United Nations University (UNU), Tokyo, Japan under the auspices of the UNESCO. In January 2001, the United Nations University set up an autonomous non- profit organization in Geneva, Switzerland to be responsible for the development and management of the UNL Program; the Universal Networking Digital Language (UNDL) Foundation[1]. 15 different languages joined the project, with each responsible for the development and maintenance of the components of its respective language module.

Since 1996, the UNL program has passed through many phases of development and enhancement and crossed important milestones (More information about the earlier system can be found in [3]; [5]; [7] and [12]. However, after 15 years of development, and after the massive project of converting parts of the Encyclopedia of Life Support Systems[2] into UNL [9] [14], many problems have surfaced and several questions about the construction and components of the UNL system have been raised.

Solutions to these problems as well as other enhancements are proposed in a comprehensive renovation project called the UNL^{+3}. UNL^{+3} does not only offer improvements to the existing infrastructure, it changes some of the core fundamentals of UNL. The change encompasses the linguistic components (Universal Words, Relations, Attributes and Features) as well as the non-linguistic ones (tools, engines and applications). Also, the structure of the linguistic resources (grammars, dictionaries, corpora..etc.) that handle these components has been drastically changed.

This paper examines the UNL^{+3} program; its ideology and architecture, and the changes and innovations it introduced to the UNL system. The scope of the program is divided into five main areas; and, thus, the paper is arranged in accordance with this division, but first, section 2 briefly introduces the UNL^{+3} program and these five areas. Then, section 3 of the paper tackles the first area; that of UNL philosophy and specifications, emphasizing the changes UNL^{+3} entailed in the linguistic components of the UNL system, starting with Universal Words (UWs), then, Relations and Attributes, and ending by discussing the Universal Tagset proposed in UNL^{+3}. Section 4 discusses the change within the second area; the linguistic resources employed by the system; the dictionaries, analysis and generation grammars, ontologies and corpora, all of which have witnessed drastic change in their ideology, structure and components and ends by introducing the new environment used to produce all of these resources; the UNL^{arium}. Section 5 approaches the third area; the engines and tools employed in analyzing and generating natural language. This section also presents the wrapper environment used to produce, develop and utilize these

[1] More information can be found at http://www.undl.org
[2] http://www.eolss.net/

tools as well as the applications that are to be based on UNL technology; the UNL^{dev}. In section 6, the paper explores the fourth area; UNL applications. UNL-based applications include the machine translation system LILY, the digital library of texts, TUT, and the knowledge extraction system, KEYS. Section 7 of the paper discusses the fifth area of interest in the UNL^{+3} program, the UNL strategy for promoting the UNL system; the UNL world. Finally, having explored the five areas of change, section 8 briefly examines some of the projects that are being carried out according to the specifications of UNL^{+3}.

2 UNL^{+3}

The UNL^{+3} is a three-year (2009-2011) plan to fulfill the initial goals of the program that previous versions failed to completely achieve and to make the UNL infrastructure fully operational by the 15th anniversary of UNL to be celebrated in 2011. UNL^{+3} should be able to advance the long-term mission of the UNDL foundation as mandated by the United Nations: to provide the UNL system world-wide for the benefit of all peoples.

The main enhancements brought about by the UNL^{+3} program can be summarized in making the UNL system more linguistic than pure engineering. As will be explained later in the paper, the older structure of the system's components -especially the tools (analysis and generation engines) and the language resources (dictionaries, grammars, etc.)- was too inflexible to accommodate the linguistic complexity of natural languages. Numerous linguistic phenomena had no place to fit in this rigid structure, and, therefore, many problems were left unsolved.

Thus, the first and most significant step in resolving such issues was the provision of the set of tools employed in the system as open-source software. Before, the source codes of the component engines and tools were protected by copyrights and, therefore, were not open for adjustments to suit the individual requirements of different languages. Hence, this new improvement allows developers to make the best use of the available tools in a way that facilitates the processing of the particular language they are working on. Similarly, the language resources (lexicons, grammars and corpora) are now also available for exporting and importing for free at the UNL^{arium} (discussed in section 4 *E*), a fact that allows the different UNL developers to share their resources and experience for the benefit of the UNL system as a whole, and, thus, facilitates the integration of new languages into the system.

A different improvement that has to do with the promotion of the UNL system is encouraging the participation of the general public in the development the system. The UNL^{wiki} is introduced, it is a collaborative website for providing and exchanging information and experience related to the UNL system. It contains all the specifications and documentation that new participants need to know about the UNL system. Browsing the UNLwiki does not require any registration or certification; it is freely accessible at (http://www.unlweb.net/wiki/index.php/Main_Page). In addition to the UNL^{wiki}, the VALERIE (VirtuAl LEaRnIng Environment) is also introduced to teach those wishing to be part of the project how to deal with natural language phenomena -especially in connection with the Universal Networking Language- in order to be actual certified developers and contribute in the construction of the system. VALERIE is free and open for the general public at (http://www.unlweb.net/valerie/).

The UNL^{+3} program is divided into five main scopes of interest, each of them covering a specific area of projects. They are:

- UNL Philosophy and Specifications: It is devoted to defining the formalisms of the linguistic components used by the UNL system such as the syntax of Universal Words; the structure of UNL documents and UNL sentences; and the set of Relations and Attributes…etc.
- UNL Resources: These are the set of linguistic and knowledge resources necessary for the analysis of natural language texts in order to form the UNL documents, and the generation of natural language out of UNL documents. The UNL system comprises four different types of language resources: lexica, grammars, ontologies and corpora.
- UNL Engines and Tools: These are the back-end software developed to perform UNL-based tasks such as analysis, generation, knowledge extraction, etc. and to assist linguists in producing UNL resources.
- UNL Applications: These are the front-end software based on UNL technology. They are designed to allow end users to accomplish natural language processing tasks, such as translating, summarizing, rephrasing, retrieving or extracting information.
- UNL World: This category involves the overall strategy for disseminating the knowledge on UNL, and for UNL promotion.

Figure 1 shows the general architecture of the UNL system according to the new specifications of UNL^{+3}, and the stages and tools involved in transforming a natural language text into a UNL document, and then into a natural language text in a different language. Each of the components shown will be subsequently discussed in the paper.

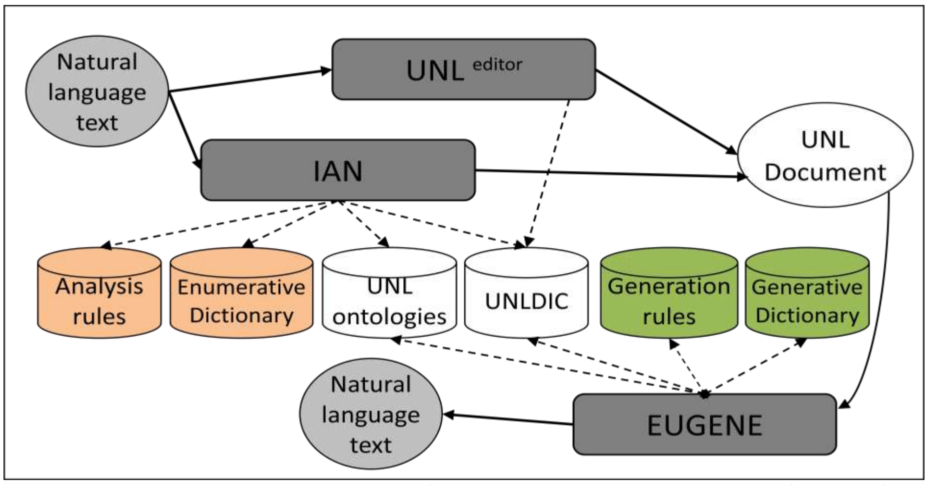


Figure 1: The overall architecture and components of the UNL system according to the specifications of UNL^{+3}

3 UNL Philosophy and Specifications

This area of the UNL system has been subject to drastic change at the hands of the UNL^{+3} plan. The change encompassed the logic and structure of the Universal Words, and the set of Relations and Attributes. Innovations, on the other hand, include introducing a universal tagset of the linguistic features to be utilized in the language resources of the system.

A. Universal Words (UWs)

Universal Words, or UWs, constitute the vocabulary of the Universal Networking Language. They are labels that stand for abstract language-independent units of knowledge (concepts) belonging to any of the open lexical categories (nouns, verbs, adjectives and adverbs). However, unlike natural language vocabulary, UWs should be free from any form of ambiguity, and - as implied by their nomenclature- universal and not biased to any particular language or culture. And for processing purposes, UWs should also be as efficient as possible.

Unfortunately, these criteria were not sufficiently fulfilled in the previous versions of UNL. In earlier versions, UWs were composed of an English headword written in Latin characters, and a constraint list that indicates the intended meaning of the headword, also written using English words. On the other hand, the new representation of UWs as specified in the UNL^{+3} program is a numerical ID that stands for the exact sense of the concept. Figure 2 shows the old and new UW representations of the concept “a writing implement with a point from which ink flows”. In the figure “poor” is the headword while “(aoj>thing)” is the constraint list.

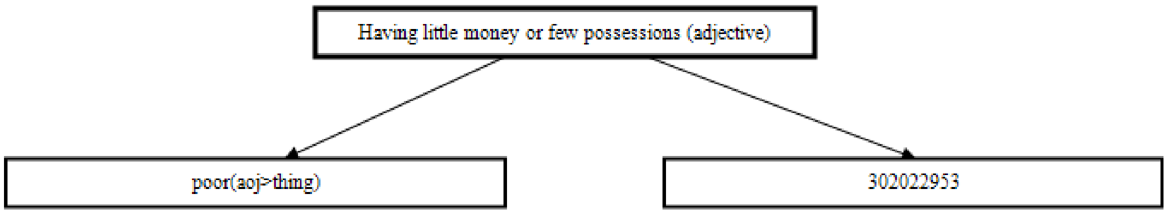


Figure 2: The older UW representation of the above concept (left) and the new UW representation (right)

Clearly, the older representation did not fulfill the unambiguity criteria; for example, it could not differentiate between "poor" the adjective describing lack of money or wealth, "poor" meaning miserable or sad, or even “poor” meaning unsatisfactory since all would have the same adjectival representation “poor(aoj>thing)”. As unambiguity is the main characteristic of UWs that sets the Universal Networking Language apart from natural languages, UNL^{+3} has set out to confront this problem with the help of the English WordNet 3.0. The WordNet is a large lexical database of English developed at the Cognitive Science Laboratory of Princeton University, it is widely used in the fields of computational linguistics and natural language processing [16]. In the WordNet, nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms (synsets), each expressing a distinct universal concept and each assigned a distinct ID number, a gloss, examples and a frequency number. The new representation, thus, has used these synsets, glosses and, more importantly, IDs to differentiate between even the slightest nuances in meaning. Thus, “poor” with the three previous meanings would be represented by the following three IDs in table 1, respectively.

TABLE I
TWO WORDNET IDS REPRESENTING THREE DIFFERENT SENSES OF THE WORD "POOR"

ID Number:	Synset:	Gloss:	Examples:	Frequency:	Part of Speech:
302022953	Poor	having little money or few possessions	"deplored the gap between rich and poor countries", "the proverbial poor artist living in a garret"	12	ADJ
301050890	hapless, miserable, misfortunate, pathetic, piteous, pitiable, pitiful, poor, wretched	deserving or inciting pity	"piteous appeals for help", "pitiable homeless children", "a pitiful fate", "Oh, you poor thing", "his poor distorted limbs", "a wretched life"	24	ADJ
301128719	Poor	unsatisfactory	"a poor light for reading", "poor morale", "expectations were poor"	0	ADJ

As for Universality, in the older phase of UNL development, UWs were represented in the form of an English headword modified by an English constraint list that restricted its meaning to a particular universal sense. Evidently, this representation with its use of English words to stand for a universal concept severely undermined the supposed universality of UWs and, consequently, of UNL. Luckily, UNL^{+3} has put an end to this issue by employing a representation made entirely of numbers, and, hence, free from any bias to one language or another. Using such a representation, the concept referring to “solid-hoofed herbivorous quadruped domesticated since prehistoric times” that would have been previously defined using English words as “horse(icl>mammal)” is now represented as “102374451”

We then come to the third criteria of efficiency. The older UW representation suffered from many redundancies that rendered it inadequate. One form of redundancy was due to the fact that synonymous words had to be defined independently. For example, the words “execution”, “carrying out” and “implementation” were represented by three distinct UWs, although, in fact, they share a single abstract conceptual sense equivalent to the gloss “the act of accomplishing some aim or executing some order”. In UNL^{+3}, this redundancy has been eliminated since -as mentioned earlier- each ID extracted from the WordNet is assigned to a set of Headwords (a synset) that this ID as well as the gloss and examples apply to. Hence, the three UWs that were previously used to define this concept are now unified into a single UW as shown in figure 3.

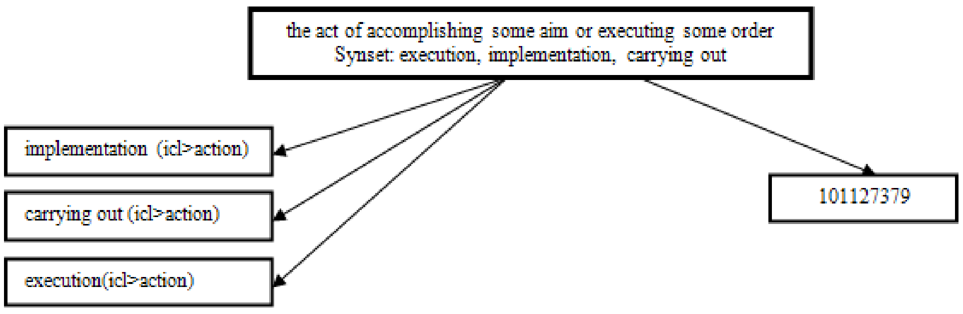

the act of accomplishing some aim or executing some order
Synset: execution, implementation, carrying out
implementation (icl>action)
carrying out (icl>action)
execution(icl>action)
101127379

Figure 3: The older three UWs previously used to represent the above concept (left) and the new combined representation (right)

Another form of redundancy also remedied by WordNet IDs was a result of the lack of consistency and conformity in the definition of UWs. Different developers would define the same exact concept differently, and, thus, the same concept would be listed in the dictionary multiple times. For example, the concept "a writing implement with a point from which ink flows" referred to above could have been defined using any of the equivalent UWs shown in figure 4 (left). On the contrary, adopting the new form of UW representation, only one ID would be applicable to this sense and, thus, such redundancy would be eliminated.

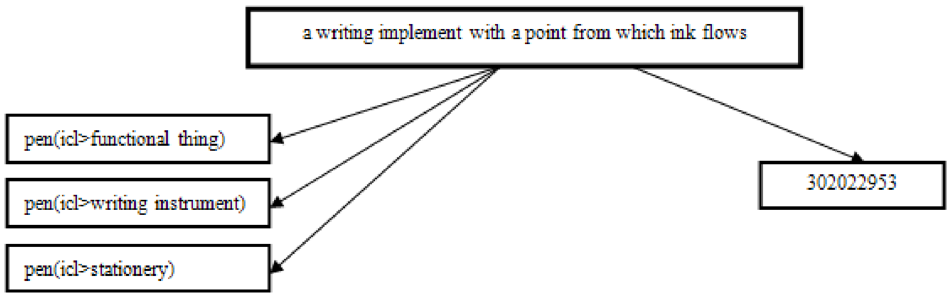

a writing implement with a point from which ink flows
pen(icl>functional thing)
pen(icl>writing instrument)
pen(icl>stationery)
302022953

Figure 4: The older three equivalent UWs previously used to represent the above concept (left) and the new unified representation (right)

This form of inconsistency, at any rate, can be ascribed to personal differences in opinion and the lack of clear rules for defining UWs. However, due to the numerous problems with the older UW representation, a more systematic kind of inconsistency started to take shape. Some of the language centers spread around the world decided to come up with solutions to these problems and devised new ways to define their UWs. For instance, the Russian center proposed to resolve the ambiguity of Universal Words by adding an extra piece of information to the constraint list. For example, it suggested defining the adjective poor as "poor(aoj>thing, ant>rich)" to stand for the antonym of rich or the first sense in table 1. On the other hand, following the same strategy, the second sense in table 1 would be defined as "poor(aoj>thing, syn>miserable)" which is the synonym of "miserable". The proposition of the Russian center was declined, however, it warned against the looming emergence of UNL dialects. Therefore, it was crucial that UNL^{+3} formulate an approach for defining UWs that is both efficient to suppress any redundancies, and consistent to guarantee that it be easily followed by the participant individuals and institutions; hence was the adoption of WordNet-based Universal Words.

More information about the older representation of Universal Words is found in [2] and at (http://www.undl.org/publications/UW%20and%20UNLKB.htm). However, [4], [6], [10] and [15] discuss using the WordNet in developing UNL lexicons. Further explanation of the new UW format adopted in UNL^{+3} can be reached at (http://www.unlweb.net/wiki/index.php/Universal_Words).

B. Relations

Relations are the semantic tags used to determine the kind of relationship between two-and only two- UWs in the UNL graph. They may represent semantic cases or thematic roles such as agent, object, instrument, etc. Although UNL Relations might occasionally coincide with syntactic relations, UNL Relations -as they were first invented in 1996- are never meant to denote syntactic or grammatical categories; they rather stand for the abstract conceptual one-way link between the words in the source natural language sentence and, consequently, the UWs in the UNL graph. This, of course, has made the choice of the appropriate Relations one of the greatest challenges in constructing semantic networks in case of analysis, and generating intelligible sentences out of a semantic network in case of generation.

The set of Relations to be used in UNL grammars is defined in the UNL specifications and is not open to frequent additions. It did not undergo as drastic a change as the UWs; UNL^{+3} mainly adopted the set of Relations defined in the latest version of UNL specifications [2]. However, the UNL^{+3} program has organized Relations into a hierarchy where lower nodes inherit the properties of upper ones. The hierarchy includes four general categories: participant (ptp) for the necessary arguments of verbal concepts such as subjects and complements; attribute (aoj) for the necessary arguments of nominal predicates such as subjects and complements; specifier (mod), for general specifiers; and adjunct (adj), for general adjuncts, including time, location and manner. Figure 5 shows the hierarchy of adjunct relations as organized in UNL^{+3} specifications.

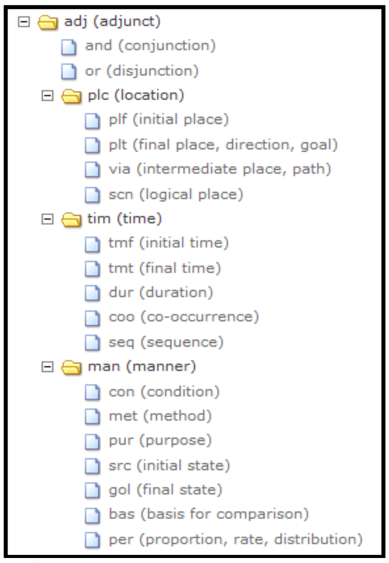

adj (adjunct)
and (conjunction)
or (disjunction)
plc (location)
plf (initial place)
plt (final place, direction, goal)
via (intermediate place, path)
scn (logical place)
tim (time)
tmf (initial time)
tmt (final time)
dur (duration)
coo (co-occurrence)
seq (sequence)
man (manner)
con (condition)
met (method)
pur (purpose)
src (initial state)
gol (final state)
bas (basis for comparison)
per (proportion, rate, distribution)

Figure 5: The hierarchy of adjunct (adj) Relations as set in the UNL^{+3} specifications

Another change brought about by UNL^{+3} to the set of Relations is bringing back into use some Relations that had been suspended in older specifications. For example, the Relation "DUR" for "duration" has been re-employed as it proved essential in sentences such as "He wanted a rope to tie the horse during the day" where the two concepts [tie] and [day] must be linked via a Relation that denotes the period of time in which the main entry existed or persisted. On the other hand, UNL^{+3} has dispensed with some Relations that proved redundant such as the relation "PPL" for Physical Place which was found to be useless in the presence of the relations "PLC" for "place" and "SCN" for "scene", where the first is used for physical places and the second for immaterial places. The specifications of 2005 still apply in UNL^{+3} [2], however, more explanation of Relations within the framework of UNL^{+3} can be found at (http://www.unlweb.net/wiki/index.php/Relation).

C. Attributes

Attributes are additional tags that encode the contextual and/or subjective knowledge present in the original sentence into the UNL graph. They are used to further modify the semantic network and add information that is not expressed via UWs or Relations. The main drawback in the older set of Attributes was its insufficiency to represent all the information conveyed in the natural language sentence. Hence, the UNL^{+3} program has augmented the set of Attributes and classified them into three different categories:

- Attributes conveying information on the role of the node in the UNL graph such as the Attribute "@entry" that indicates the main node in a UNL graph;
- Attributes conveying information on the original co-text of the node it modifies such as the Attribute "@parenthesis" that indicates that the node was originally put between parentheses; and
- Attributes conveying information on the (external) context of the utterance such as the attribute "@past" that indicates that the node was used in a time before the time of the utterance.

The set of Attributes is defined in the UNL tagset and is not open to further additions. Similar to UWs, Attributes have undergone rigorous changes, both in quality and quantity. Concerning quantity, the number of Attributes has quadrupled as UNL^{+3} tried to remedy the deficiencies in earlier UNL specifications. UNL^{+3} introduced some indispensible Attributes, such as those denoting gender. Before, such a crucial piece of information was lost in the conversion of natural language into UNL. For example, the UNL semantic network for the French phrase "Mémoires d'un médecin" "Diaries of a physician" did not indicate whether the concept [médecin] is masculine or feminine although in the French original it is clearly masculine (preceded by the French masculine indefinite article "un"). Thus, on trying to generate the Arabic equivalent of such a network, it was not clear whether it should be "مذكرات طبيب" or "مذكرات طبيبة".

In addition to those of Gender, many other categories of Attributes such as those denoting animacy (@person and @thing), degree (@more, @equal, @least, etc.), figures of speech (@ellipsis, @metonymy, @euphemism, etc.) and many others that are crucial in representing the full meaning of natural language utterances have also been introduced in the UNL^{+3} program. The uppermost categories (old and new) in the Attribute hierarchy are shown in figure 6.

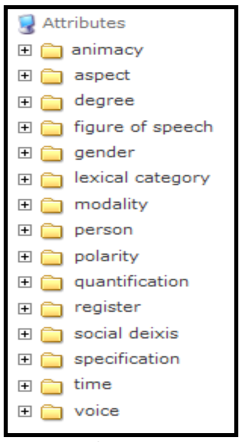
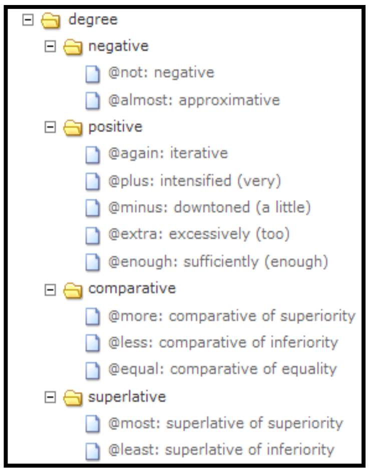


Figure 6: The uppermost categories of Attributes as specified in the UNL^{+3} tagset

In addition to these novel categories, UNL^{+3} has also augmented the older categories with new Attributes. For example, the attributes “@inceptive”, “@prospective” have been added to the category of Aspect and the Attributes “@paucal” and “@multal” have been added to the category of Number. Moreover, some of the existing categories have been divided into a finer classification such as the category of Degree which is now divided into the subcategories: negative, positive, comparative and superlative as shown in figure 7.



degree
negative
@not: negative
@almost: approximative
positive
@again: iterative
@plus: intensified (very)
@minus: downtoned (a little)
@extra: excessively (too)
@enough: sufficiently (enough)
comparative
@more: comparative of superiority
@less: comparative of inferiority
@equal: comparative of equality
superlative
@most: superlative of superiority
@least: superlative of inferiority

Figure 7: The finer classification of the Attribute category of degree in UNL^{+3}

On the other hand, the quality of older Attributes have been enhanced by classifying the set of Attributes into the taxonomic hierarchy shown in figures 6 and 7 so that lower values may assume upper ones.

Another reason behind this substantial increase in the set of Attributes is that many natural language words that were previously treated as distinct concepts are now reconsidered. For instance, all closed classes of lexical items (demonstratives, adposition, conjunctions, pronouns, etc.) are now represented as Attributes rather than concepts since, in reality, they do not identify an abstract universal concept, they simply link between concepts or refer to ones previously identified. Demonstratives are now represented via the Attributes “@proximal”, “@medial”, etc. while adpositions are represented by Attributes such as “@under”, “@along”, “@after”, etc. Similarly, the Attributes “@before”, “@and”, “@since” and many others represent the set of conjunctions. As for pronouns, they are denoted by the Attributes “@1” for first person, “@2” for second person or “@3” for third person while their gender is denoted by the Attributes “@male” or “@female” and their number is denoted by the Attributes “@singular”, “@dual”, “@plural”, etc. For example, the pronoun “they” is represented by the attributes “@3”, “@male” and “@plural”.

Other words that were previously identified as concepts include the words “each other” in a sentence like “They comforted each other” which is now denoted by the Attributes “@each” and “@other”. Another example is the interrogative word “which” in the question “which book are you reading?”, this too is signified by the attribute “@wh”.

More examination of the set of Attributes in older UNL versions is available at the UNL Specifications 2005 [2] while the UNL^{+3} set of Attributes and their uses can be found at (http://www.unlweb.net/wiki/index.php/Attribute).

D. Tagset

Besides UWs, Relations and Attributes, UNL-based dictionaries and grammars also require a metalanguage capable of accurately describing the linguistic behavior of words and structures. This metalanguage is composed of a diverse set of linguistic attributes and values that are either inserted in the dictionary along with the entries to indicate a word's part of speech, number, valency, etc.; or used in analysis and generation rules to indicate a sentence's agreement, voice, distribution, etc. However, the required linguistic features and their values may vary drastically from one language to another depending on the nature and the structure of the language under consideration and, in fact, they did in earlier versions of UNL. There was no fixed set of linguistic features, or values; each language center created its own list of features and values, and chose how to use it in defining their UWs and building their grammars, a fact that hampered the exchange of language resources among the participant language centers, and undermined their understadability.

Hence, in order to better standardize the language resources inside the UNL framework (dictionaries, grammars, corpora, etc.), UNL^{+3} laid down a sole universal tagset capable of describing the pervasive morphological, semantic, syntactic and even pragmatic phenomena, as well as the specific ones peculiar to only a handful of languages. Each language center can, then, choose the set of values applicable to the phenomena the language manifests. For instance, the UNL tagset covers all possible values of countability; singular, dual, trial and quadral, paucal, multal, plural, singulare tantum, plurale tantum and invariant although the value of "dual", for example, would never be used in languages such as English or French; however, it will definitely be used in the Arabic language.

The UNL^{+3} tagset was designed in order to be as comprehensive, few, short and mnemonic as possible in order to ensure comprehensibility and consistency. It standardizes both: the tags of the linguistic attributes, and the tags of the values they may assume. For example, UNL^{+3} postulates that the linguistic attribute of "number" must be signified by the tag "NUM" and that its value may only be represented by one of the tags shown in figure 8. Figure 8 also shows that the set of linguistic attributes and their values are arranged in a taxonomic hierarchy so that upper level values could be inferred from lower ones.

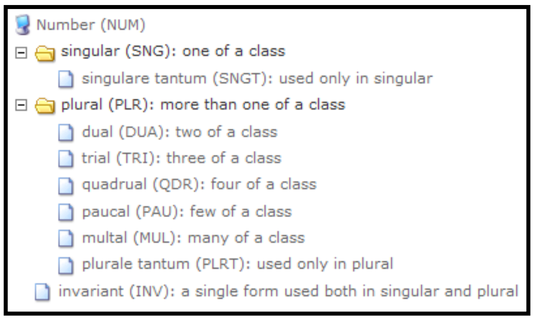

Number (NUM)
- singular (SNG): one of a class
  - singulare tantum (SNGT): used only in singular
- plural (PLR): more than one of a class
  - dual (DUA): two of a class
  - trial (TRI): three of a class
  - quadrual (QDR): four of a class
  - paucal (PAU): few of a class
  - multal (MUL): many of a class
  - plurale tantum (PLRT): used only in plural
- invariant (INV): a single form used both in singular and plural

Figure 8: The linguistic attribute of "number" and its possible values as set by the UNL^{+3} unified tagset

Such standardization is crucial in ensuring full translatability across the languages participant in the UNL program; generation grammars can consult the universal tagset whenever they come across a value that does not exist in their native language. In addition, standardizing the set of tags across the UNL community facilitates understanding and exchanging the available language resources of different languages among the various UNL language centers. Figure 9 shows the linguistic attributes (uppermost level) in the tagset hierarchy introduced in UNL^{+3}. The UNL^{+3} tagset can be reached at (http://www.unlweb.net/wiki/index.php/Tagset).

4 UNL RESOURCES

Not only have the linguistic components employed in the UNL infrastructure been subject to change, the structure of the language resources that organize and employ such components have also been the subject of dramatic improvements on the hands of UNL^{+3}. Language resources are required to perform the various UNL-based tasks such as translation, summarization, knowledge extraction, etc. They include dictionaries, analysis and generation grammars, ontologies and corpora.

A. Dictionaries

UNL-NL and NL-UNL dictionaries are bilingual dictionaries where lexical items of a given natural language are matched with their corresponding abstract language-independent Universal Words (UWs). They are the cornerstone of any UNL-based application as they contain all the information required to perform successful analysis (UNLization) and generation (NLization) processes. Unfortunately, in the older UNL version, dictionaries suffered from numerous problems that prevented it from playing its role in fulfilling the ultimate goals of UNL. The problems were mainly related to the structure of the dictionaries and the way entries are stored in them. Therefore, it was only natural that UNL^{+3} introduce some drastic changes to the structure of dictionaries, all of which are meant to improve their efficacy and potential.

Probably the most significant change introduced by UNL^{+3} was disuniting the generation (UNL-NL) and analysis (NL-UNL) dictionaries. Although at first sight this modification might seem to reduce efficiency rather than boost it, in reality, it is a major improvement. This disunion came as a result of changing the way natural language headwords are stored in the generation dictionary, which is another modification brought about by UNL^{+3}. Following the specifications of UNL^{+3}, generation dictionaries should (i.e. UNL-NL) only store a single base form for the lexical item. This is because UNL^{+3} specifications now allow dictionary developers to write the linguistic rules capable of generating all the possible word forms of the stored base form along with each entry in the generation dictionary. These rules can be affixation rules (A-rules) used for prefixation, suffixation and infixation, or linear rules (L-rules) used for applying transformations over ordered sequences of isolated words such as generating spelling changes (contraction, elision, assimilation, etc.), the use of capital letters and punctuation marks.

However, this strategy is not applicable to the analysis dictionaries as it would be quite time-consuming to predict the base form of the incoming source language word forms. Hence, analysis dictionaries must still independently store the stems representing all the word forms of a lexical item, thus, the analysis and generation dictionaries had to be disunited. This composite amendment has improved the dictionary efficiency by eliminating the redundancy of storing the same concept in several forms when they are not needed; i.e. in generation, and, thus, has speeded up the processing time in both the analysis and generation processes.

In addition to inflectional rules, UNL^{+3} has also allowed the insertion of subcategorization rules along with the entries in generation dictionaries. Subcategorization rules describe the syntactic behavior of the word and are responsible for determining the number and types of the necessary syntactic arguments (specifiers, complements and adjuncts) that co-occur with the entry in order to form a multi-word expression or a phrase; i.e., its maximal projection. The verb "give", for instance, generally requires at least one specifier (the subject) and two objects (a direct and an indirect), even if in several contexts they are not explicit.

Generally, both Analysis and Generation dictionaries follow the format shown in figure 9.

```
[NLW] {ID} "UW" (ATTR , ... ) < FLG , FRE , PRI >;
```

Figure 9: The general format of generation and analysis dictionary entries

Where:

NLW is the headword of the natural language lexical item.
ID is the unique identifier of the entry.
UW is the abstract concept representing the natural language word.
ATTR is the list of linguistic features of the NLW; these are set according to the UNL tagset. It also includes the affixation, linear and subcategorization rules in the generation dictionary.
FLG is the three-character language code according to ISO 639-3[3].
FRE is the frequency of NLW in natural texts. It is used in natural language analysis (NL-UNL).
PRI is the priority of the NLW. It is used in natural language generation (UNL-NL).
COMMENT is any comment necessary to clarify the mapping between NL and UNL entries.

The format of dictionary entries is not different from the one used in earlier UNL versions, however, the structure of the NLW, UW, ATTR components has witnessed some change. The UW change has been discussed before in section 3A, and the change in ATTR component is the integration of affixation, linear and subcategorization rules in the generation dictionary discussed earlier in this section. The remaining change is the one involving the NLW. The change has to do with the types of

[3] http://en.wikipedia.org/wiki/List_of_ISO_639-2_codes

headwords (NLWs) that can be stored in the dictionary. Before, a natural language entry could either be stored in the form of a simple headword such as “book”, a multiword headword such “The United States of America”, a compound headword such as “socio-economic or even a simple morpheme. Although there were several options, they all entailed serious limitations; all of the previous forms were treated as a single unit and infixation was by all means impossible. Therefore, UNL^{+3} introduced two new forms of NLWs; namely, complex structures and regular expressions.

Complex structures are introduced in order to deal with discontinuous multiword lexical items such as “take into account”. Using a complex structure, the natural language NLW can be represented as a complex structure made of several sub-NLWs, each sub NLW can have a different part of speech and a distinct set of linguistic attributes. Thus, in the previous example, [[take] [into][account]] is the NLW while [take], [into] and [account] are the sub-NLWs. This new form allows for the very crucial option of inserting a lexical element between the constituents of a discontinuous lexical item to generate, for example, “taking the previous into account”. Previously, dictionary developers did not have that option, they could either consider the discontinuous concept a compound, in which case it would be treated as single unit and inserting any lexical unit between its constituents would be unfeasible, or only choose a part of the concept to be the NLW and try to attach the other part to it by means of some features and attributes; both options were quite inefficient and impractical. The general format of complex structures is shown in figure 10:

```
[[sub-NLW][sub-NLW]...[sub-NLW]] {ID} “UW” (ATTR , ..., #01(ATTR, ...), #02(ATTR,
...), ...) <FLG , FRE , PRI >; COMMENTS
```

Figure 10: The general format for complex structure NLWs

Where:

[sub-NLW] is a part of the NLW;
#01(ATTR, ...) are the specific features for the first sub-NLW to appear in the NLW;
#02(ATTR, ...) are the specific features for the second sub-NLW to appear in the NLW; and so on.
The features outside the sub-NLW feature lists are shared by all sub-NLWs.

The second type of NLW introduced by UNL^{+3} is that of Regular Expressions. Both the NLW and the UW fields may be represented in the form of regular expressions. In both cases, regular expressions must be included between a pair of "/" and should comply with the PCRE – Perl Compatible Regular Expressions[4]. In the field of NLW, Regular expressions are useful in eliminating redundancy. For example, rather than including two entries in the dictionaries to represent “color” and “colour” which are different spelling conventions for the exact same concept, the NLW can be represented as the regular expression [/colo(u)?r/] which means that the NLW is either equal to the string "color" or "colour". Similarly, according to the UNL^{+3} specifications for regular expressions, the regular expression [/(\d){4}/] means that the NLW is any sequence of four digits (quite useful in indicating years).Figure 11 shows the genera format for a dictionary entry in which the NLW is a regular expression.

```
[/<RegEx>/ “<UW>” (<FEATURE LIST>) <FLG,FRE,PRI>]
```

Figure 11: The general format of regular expressions in the NLW field

In the similar manner, UWs can be represented as regular expressions; for example, the regular expression "/(.)+\(iof\>city\)/" means that the UW is any one ending by the string "(iof>city)". This is especially useful in the generation process. The general forma for a dictionary entry in which the UW is a regular expression is shown in figure 12.

```
[<NLW>] “/<RegEx >/” (<FEATURE LIST>) <FLG,FRE,PRI>]
```

Figure 12: The general format of regular expressions in the UW field

Further discussion of the structure of dictionaries in older UNL versions is found in [2] while detailed examination of the dictionary specifications according to the UNL^{+3} program is found at (http://www.unlweb.net/wiki/index.php/Dictionary_Specs).

B. Analysis and Generation Grammars

UNL grammars are the set of rules responsible for analyzing or composing natural language sentences and ensuring their well-formedness. They govern the composition of sentences, phrases and words in any given natural language. They are generally

[4] http://www.pcre.org/

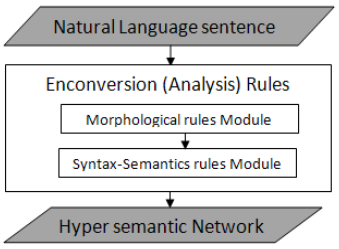
unidirectional; from natural language into UNL (UNLization or analysis grammars) and from UNL into natural language (NLization or generation grammars).

The UNL^{+3} program did not only change the components and structure of the UNL grammars, it drastically altered the ideology and linguistic infrastructure on which these grammars are based. As mentioned earlier, the design of the grammars in previous UNL versions was mainly engineering rather than linguistic and the various linguistic phenomena were forced to fit into that engineering design. As a result, the older UNL grammars did not employ a particular linguistic theory in defining their components and devising their processes. Thus, in order to maintain consistency and comprehensibility across the grammars of all languages, UNL^{+3} chose as a linguistic foundation an internationally-acknowledged linguistic theory; namely, the X-bar theory [8] and [11].

Another drastic change is in the processes of analysis and generation themselves. In the older generation of UNL, the original grammar design was not broken up to accommodate the different categories of linguistic phenomena; instead, only one module was provided by the system. In this module, grammar developers (computational linguists) organized their grammars in a way that facilitates handling the language they aim to process. This, of course, undermined the modularity of the system, and reduced the efficiency of the rules and accuracy of the output. Therefore, UNL^{+3} opted for a more linguistic approach by dividing the grammar into seven types of rules, each devoted for resolving a specific type of linguistic issues. Three of these types are common between analysis and generation, the other four are more specific; two for analysis and two for generation. These types of rules and their functions will be discussed thoroughly in the following subsections 1 and 2 of this section.

These seven types constitute the main body of rules that are responsible for analyzing natural language into UNL semantic networks (the UNL graph) and generating natural language out of them; i.e. transformation rules. However, another equally important category of rules are those devoted to disambiguation. Previously, disambiguation was handled by a type of rules called backtrack rules. Backtrack rules constituted one of the most deep-seated shortcomings in former UNL versions. First, they were only meant to handle lexical ambiguity; they could not resolve syntactic or structural ambiguity. Second, they were not efficient in resolving lexical ambiguity and never fully performed their task. Hence, in UNL^{+3}, backtrack rules are replaced by disambiguation rules. Disambiguation rules are used to restrict the applicability of transformation rules by assigning priorities in order to prevent wrong lexical choices, to prompt best matches or to check the consistency of the graphs, trees and lists. There are three types of disambiguation rules that can handle ambiguity on the level of semantic networks, syntactic trees and natural language lists. These too will be discussed in more detail in the following subsections 1 and 2.

1) *Analysis grammar:* as mentioned earlier, in the older version of UNL, analysis rules were bundled up in a single module. Thus, the process of analyzing natural language into semantic networks was done over a single leap. Later on, it became clear that such a non-linguistic approach will not be sufficient in dealing with the complex phenomena manifested by natural languages, and will not be enough to disengage the interweaving of syntax, semantics and morphology in natural language sentences. Therefore, according to the specifications of UNL^{+3}, analysis grammar is to be more modular. It uses five of the seven types of transformation rules mentioned above. These five rule types represent the five stages through which input sentences pass starting from natural language sentences passing through syntactic trees and finally forming the semantic network. Figure 13 illustrates the older design of analysis grammar showing the one-step process. By contrast, figure 14 illustrates the more elaborate design of the analysis grammar as postulated by the UNL^{+3} program showing the five stages of analysis.


Natural Language sentence
Enconversion (Analysis) Rules
Morphological rules Module
Syntax-Semantics rules Module
Hyper semantic Network
Figure 93: The analysis grammar design in the older UNL system

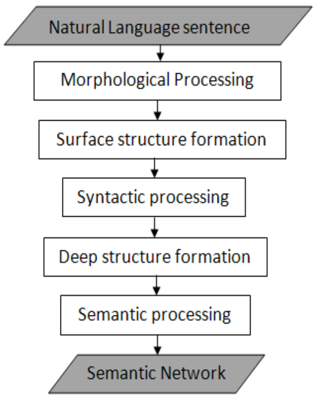


Figure 14: The new analysis grammar design according to the specifications of UNL^{+3}

The five analysis phases are:

1) LL - List Processing (List-to-List)
2) LT - Surface-Structure Formation (List-to-Tree)
3) TT - Syntactic Processing (Tree-to-Tree)
4) TN - Deep-Structure Formation (Tree-to-Network)
5) NN - Semantic Processing (Network-to-Network)

The List to List (LL) rules are responsible for preprocessing the natural language input by analyzing it morphologically in order to match the input words with the dictionary entries and assign each stem to the concept it conveys. For example, in this stage, the word “boys” is analyzed into “boy” + “s”. Consequently, “boy” is matched with the concept “110285313” while the suffix “s” is deleted and replaced by the attribute “@def” which is attached to the stem of the word.

Then, List-to-Tree Rules (LT) parse the resulting list structure into a surface tree structure. This type of rules is only employed in the analysis process. They specify the syntactic relations between the words of the input sentence to form a surface tree structures.

Third, Tree-to-Tree Rules (TT) are used for revealing the deep structure underlying the previously formed surface structure. In this phase, bottom-up parsing takes place to form a higher level of linguistic description. The process starts by composing higher constituents from lower ones going from right to left. Because rules are recursive, they will pass again from right to left to form any possible larger constituent.

Fourth, Tree- to – Network Rules (TN), these form the initial semantic representation out of the deep syntactic structure. In this stage, each syntactic relation is replaced by an appropriate semantic relation; in other words, syntactic arguments are mapped with their semantic counterparts [1].

Fifth, and finally, Network-to-Network Rules (NN) are used for post-editing the semantic network structure derived from the syntactic module in order to generate the final UNL graph.

Parallel to the work of the previous transformation rules is the application of analysis disambiguation rules. When analyzing a natural language sentence, disambiguation is needed to determine which universal concept (UW) is intended by the natural language word. Naturally, many UWs are possible candidates for the same input word. The older system was deterministic, thus, backtrack rules could only choose a single UW at a time and try to constitute the semantic network using it. Backtrack rules used to match the longest string possible of the input word with dictionary entries and choose the first one it meets. The chosen candidate is then tested and, if it failed to fit into its linguistic context, backtrack rules would then go back and choose the next possible candidate in the dictionary, and so on and so forth. At many times backtrack rules would go through hundreds of candidates before reaching the suitable one, if it did. This, of course, constituted a massive processing load and, hence, had to be changed.

This issue of lexical ambiguity is, fortunately, resolved by the new disambiguation rules introduced in the UNL^{+3} program. Three types of disambiguation rules are used in analysis; network disambiguation rules, tree disambiguation rules and list disambiguation rules. List Disambiguation Rules is the type devoted to resolving lexical ambiguity; it applies over natural language list structures to constrain the application of Tree-to-List (TL) rules and select the most appropriate UW as shown in table 2.

TABLE 2

THE APPLICATION OF LIST DISAMBIGUATION RULES IN RESOLVING LEXICAL AMBIGUITY

INPUT	DICTIONARY	DISAMBIGUATION RULES	OUTPUT
the book	[book] "22222" (POS=VER); (higher priority) [book] "11111" (POS=NOU); (lower priority)	(ART)(BLK)(VER)=0;	[book] "1111" (POS=NOU);

Aside from lexical ambiguity, the other two types of disambiguation rules can also handle structural ambiguity. Tree Disambiguation Rules apply over the intermediate syntactic trees to restrict the application of List-to-Tree (LT) rules in the analysis process. For Example, the rule (VS(VER;ADJ)=0;) postulates that An adjective (ADJ) may not be an specifier (VS) of a verb (VER).

Finally, Network Disambiguation Rules apply over the UNL semantic networks to constrain the application of Tree-to-Network (TN) Transformation Rules in the process of analysis (UNLization). For example, the rule (agt(VER;ADJ)=0;) states that an adjective (ADJ) may not be an agent (agt) of a verb (VER).

By applying the previous five transformation phases as well as disambiguation rules, the final UNL graph that represents the abstract meaning of the natural language input is formed. Subsequently, this UNL graph can be translated into the natural language of choice, with the help of the target language generation dictionary and generation grammar.

For further information about the older format of rules see (http://www.undl.org/unlsys/ds.html). The new rule specifications introduced by UNL^{+3} can be reached at (http://www.unlweb.net/wiki/index.php/Grammar_Specs).

2) *Generation Grammar:* Similar to analysis, the previous UNL specifications did not categorize or classify grammar rules and the process of generating natural language sentences out of UNL semantic networks was performed in a single step. Figure 15 shows the older grammar design, and figure 16 shows the more linguistic design of the grammar according to the UNL^{+3} specifications

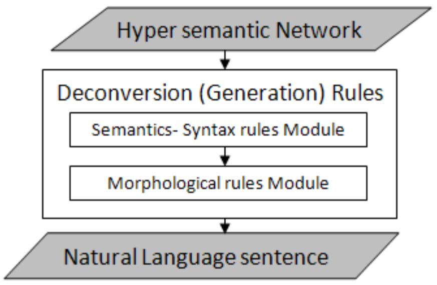

Hyper semantic Network
Deconversion (Generation) Rules
Semantics- Syntax rules Module
Morphological rules Module
Natural Language sentence

Figure 10: The generation grammar design employed in the older UNL system

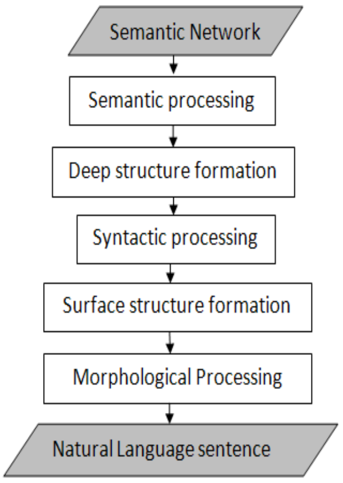


Figure 116: The new generation grammar design according to the specifications of UNL^{+3}

Thus, generation grammars are also divided into five stages that utilize five of the seven rule types employed in the system. These stages are:

1) NN - Semantic Processing (network-to-network)
2) NT - Deep-Structure Formation (network-to-tree)
3) TT - Syntactic Processing (tree-to-tree)
4) TL - Surface-Structure Formation (tree-to-list)
5) LL - List Processing (list-to-list)

First, Network-to-Network Rules (NN) are used for pre-editing the incoming UNL graph, transforming it into a semantic network that is more easily generated into the target language. For example, the English language distinguishes between attributive adjectives (nice boy) and predicative adjectives (the boy is nice); attributive adjectives precede their nouns unlike predicative ones. UNL is a universal formalism and, hence, acknowledges this distinction. It uses the "mod" semantic relation to express attributive adjectives and the “aoj" semantic relation to express predictive adjectives. On the other hand, the Arabic language does not make that distinction; adjectives always follow the modified nouns. Thus, the Arabic Network-to-Network rules transform any "mod" between a noun and an adjective into "aoj".

Second, Network-to-Tree Rules (NT) are the rules responsible for reorganizing the semantic network structure into a deep tree syntactic structure by mapping the semantic relations with their corresponding syntactic roles.

Third, Tree-to-Tree Rules are used for processing the resulting trees, transforming the deep syntactic structure from the previous stage into a surface syntactic structure. These rules gather individual syntactic relations to form higher constituents in the syntactic tree structure.

Fourth, Tree-to-List Rules are used to linearize the surface tree structure into a list structure and inserting the necessary blank spaces.

Fifth, and finally, List-to-List Rules are used for post-editing the output of the syntactic module and achieving the morphological adjustments needed to generate the final natural language sentence. This phase comprises the morphological rules that are responsible for two tasks; first, generating the final form of the target language words according to the linguistic attributes and the rules attached to each word in the UNL-NL dictionary; for example, generating the plural form of nouns and the past tense of verbs, etc. Second, these rules are responsible for achieving agreement between the nodes in the target language list structure, such as verb-subject agreement, noun-adjective agreement, etc.

Again, apart from transformation rules, disambiguation rules are used to replace backtrack rules. In the older UNL version, the generation dictionary stored all the stems of the possible word forms of the lexical item; thus, each concept had multiple natural language candidates. Backtrack rules chose the first possible candidate natural language word. Upon failure, backtrack rules would go back and choose the next possible candidate and so on. Similar to its counterpart in the analysis process, backtrack rules proved to be inadequate and inefficient.

Surprisingly, this problem of disambiguating Universal Words was resolved, not by grammar rules, but by the new specifications of the UNL-NL dictionary. Following the new specifications, the generation dictionary stores only the base form of natural language words, thus, each concept will have only one equivalent entry in the dictionary. Aside from that form of ambiguity, other forms are handled by the newly introduced disambiguation rules. Disambiguation rules provide a non-deterministic approach to resolving ambiguities. Tree disambiguation rules are used to constrain the application of Network-to-Tree (NT) rules in the NLization (generation) process, and List disambiguation rules are used to constrain the application of Tree-to-List (TL) rules. Both apply in the same manner they do in the analysis process.

After the five phases of generation and disambiguation, an output that is characterized by both accuracy and well-formedness is finally generated. The exceedingly correct syntactic composition of the output sentence can be ascribed to the fact that the system sets out from a semantic interpretation of the source language text, in the form of a semantic network, and then passes through a stage of syntactic tree formation in which the syntactic constituents of the sentence is meticulously ordered and adjusted before finally being transformed into the target language.

C. Ontologies

When world knowledge is needed to disambiguate or decipher a certain input, the UNL system refers to one of the system's Ontologies. Ontologies serve as the semantic component of the system and guarantee a more natural and accurate interpretation, and hence representation, of the original meaning, in UNL format. Three ontologies are employed in the UNL system, the UNL Ontology, the UNL Knowledge Base and the Concept Network. They are all tackled in the following two sub-sections.

1) *UNL Ontology:* The UNL Ontology is tree-like structures where UWs are interconnected through ontological relations such as icl (is-a-kind-of), and pof (is-a-part-of). The UNL ontology is not a new component in the UNL system; it has been used in earlier versions. However, in the UNL^{+3} program, a new ontology has been introduced in addition to the earlier one. The earlier one is the UW system which is a hierarchy of the UWs provided by the UNDL Foundation. The new ontology, on the other hand, is the UNL WordNet 2.1 which is a list of UWs extracted from the English WordNet 2.1.

UNL Ontologies can improve the results of the analysis (UNLization) process by helping in word sense disambiguation. As for the generation (NLization) process, the UNL Ontologies can make up for any deficiency in the UNL-target language dictionary. Figures 17a and 17b show some nominal and verbal concepts extracted from the UW system, respectively.

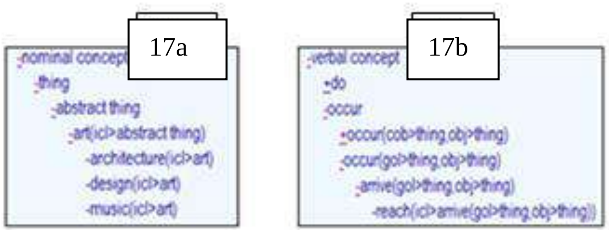


Figure 12: some nominal and verbal concepts extracted from the UW system

The UW system can be reached at (http://www.undl.org/unlsys/uw/UNLKB.htm).

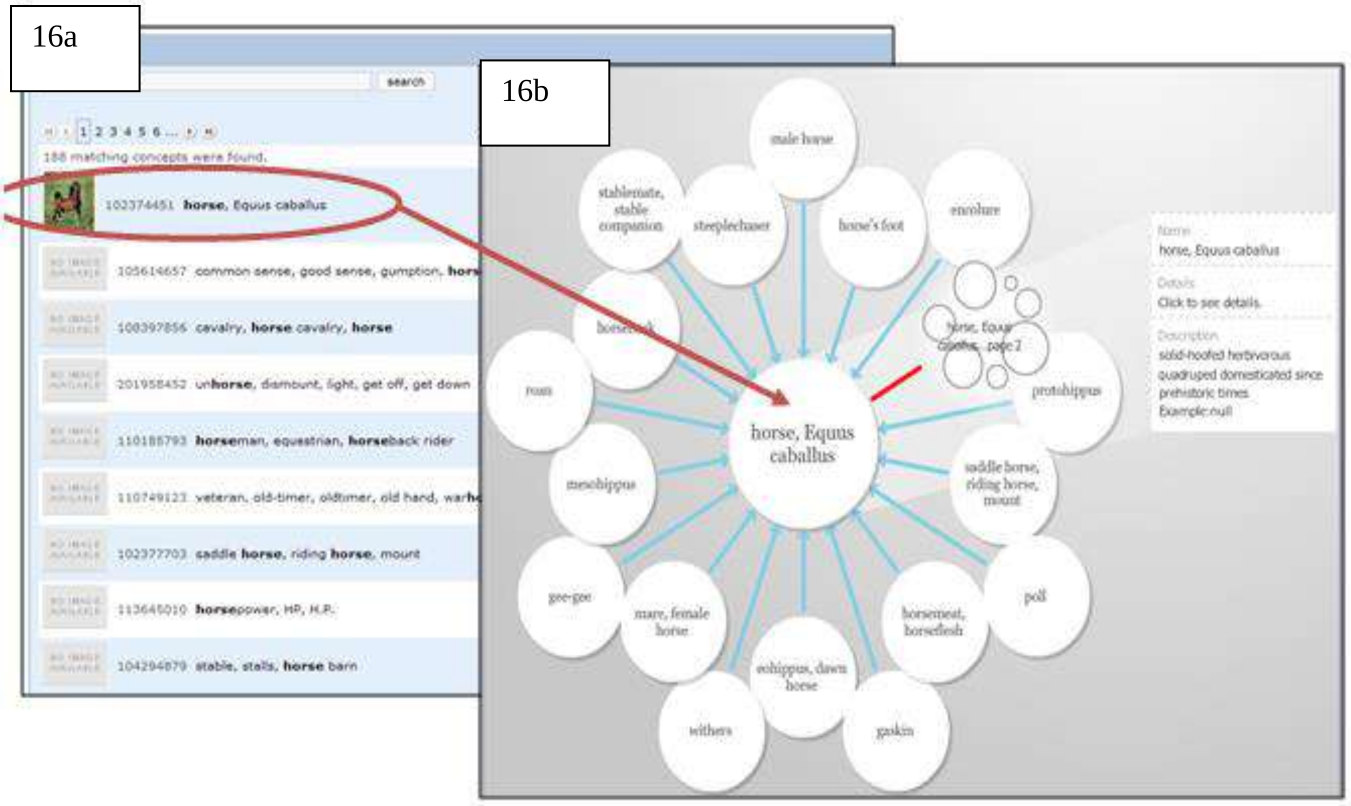
2) *UNL Knowledge Base*: The UNL Knowledge Base, or simply the UNL KB, is a newly integrated component in the UNL system. It is similar to the UNL ontology; a network structure where UWs are interlinked via any of the UNL's semantic Relations. However, the difference is that these Relations can be either ontological, or thematic such as agt (is the agent of), plc (is the place where), etc. That is to say, the UNL KB encompasses and extends the UNL ontology. Moreover, the UNL KB also attaches a degree of necessity. These and other extralinguistic information provided by the UNL KB would prove very helpful for both natural language analysis and generation processes in issues such as deciphering ambiguities, resolving anaphora and co-reference and others and, thus, enhance the capability of information retrieval and extraction through UNL.

The UNL KB is expected to be dictionary-based; representing, in UNL, the definitions of each UW extracted from several different ordinary dictionaries and from as many languages as possible, in order to best reveal the diversity and complexity of each UW. The UNL KB should be provided as XML table, as shown in figure 18.

```
<kb>
 <relation name="mod" frequency="2">
  <source id="410" attribute="entry" lang="UNL" frequency="20" class="nou">book(icl>document)</source>
  <target id="2243" lang="UNL" frequency="2" class="nou">republic(icl>form of government)</target>
 </relation>
```

Figure 13: Example from the UNL KB

3) *Concept Network:* The Concept Network is a visual ontology of concepts. It shows the semantic hierarchical relations between concepts and furnishes each with an illustrative figure, a description and other details. It is designed for creating, browsing and editing concepts and their relations. In addition, by providing a user-friendly and visually interesting overview of semantically related words, it facilitates the task of dictionary and grammar developers by putting the required concept into perspective. Figure 16a shows the list of concepts containing the word "horse" in the synset. Figure 16b is the visual network showing concepts that are semantically related to one chosen from the list.



188 matching concepts were found.

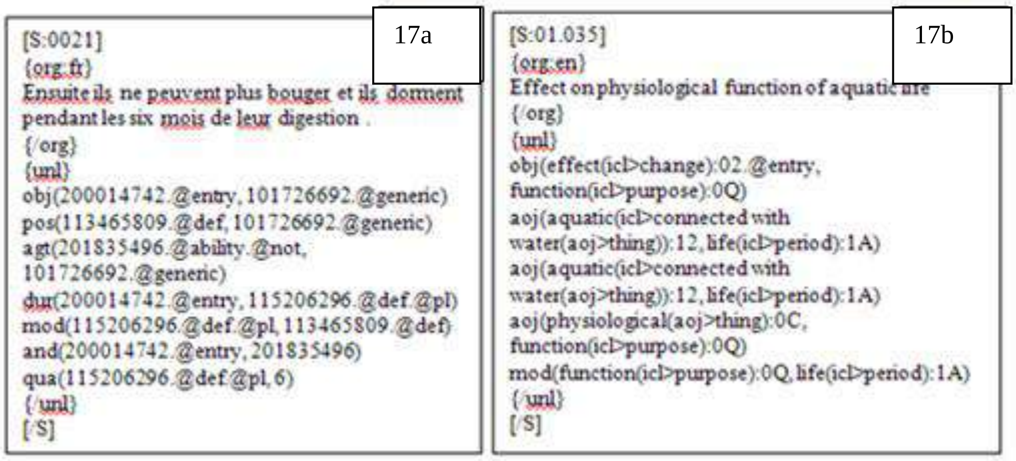
102374451 **horse**, Equus caballus
105614657 common sense, good sense, gumption, **hors**
108397856 cavalry, **horse** cavalry, **horse**
201958452 un**horse**, dismount, light, get off, get down
110185793 **horse**man, equestrian, **horse**back rider
110749123 veteran, old-timer, oldtimer, old hand, war**h**
102377703 saddle **horse**, riding **horse**, mount
113645010 **horse**power, HP, H.P.
104294879 stable, stalls, **horse** barn

Name
horse, Equus caballus
Details
Click to see details.
Description
solid-hoofed herbivorous quadruped domesticated since prehistoric times.
Example:null

Figure 16: The concept network for the synset "horse, Equus caballus"

D. UNL Corpora

Aside from the lexica, grammars and ontologies, a large corpus of the previously UNLized texts have been compiled. Corpora is not a novel component in the system, they have always been part of the UNL infrastructure even in earlier versions. They are a collection of documents written in UNL according to the UNL document structure. Corpora are mainly used for extracting entries for the UNL Example Base (discussed in the subsection 2 of this section) and the UNL Memory Base (discussed later in subsection 5 *A*). They are also used to set the standards of UNL and test the engines and tools as well as the applications to be based on UNL technology. Figures 17a and 17b show two excerpts from Le Petit Prince corpus (discussed later in 8 *D*) and the UNL-Encyclopedia of Life Support Systems corpora, respectively.


17a

```
[S:0021]
{org:fr}
Ensuite ils ne peuvent plus bouger et ils dorment
pendant les six mois de leur digestion .
{/org}
{unl}
obj(200014742.@entry,101726692.@generic)
pos(113465809.@def,101726692.@generic)
agt(201835496.@ability.@not,
101726692.@generic)
dur(200014742.@entry,115206296.@def.@pl)
mod(115206296.@def.@pl,113465809.@def)
and(200014742.@entry,201835496)
qua(115206296.@def.@pl,6)
{/unl}
[/S]
```

17b

```
[S:01.035]
{org:en}
Effect on physiological function of aquatic life
{/org}
{unl}
obj(effect(icl>change):02.@entry,
function(icl>purpose):0Q)
aoj(aquatic(icl>connected with
water(aoj>thing)):12,life(icl>period):1A)
aoj(aquatic(icl>connected with
water(aoj>thing)):12,life(icl>period):1A)
aoj(physiological(aoj>thing):0C,
function(icl>purpose):0Q)
mod(function(icl>purpose):0Q,life(icl>period):1A)
{/unl}
[/S]
```

Figure 14: Excerpts from Le Petit Prince corpus and the UNL-EOLSS corpora, respectively

The corpora of the ongoing projects can be reached via the development environment; the UNL^{arium}, at (http://www.unlweb.net/unlarium/index.php?corpus=new) for logged in users. Older corpora, however, are available at (http://www.unlweb.net/wiki/index.php/Corpora).

4) *UNL Example Base*: The UNL Example Base (UNL^{eb}), also known as the UNL Encyclopedia contains semantic relations between UWs along with a degree of probability. It is built automatically by analyzing large corpora, thus, it comprises information that is related to the probability of occurrence rather than the possibility of occurrence. It is the only component in the UNL^{arium} framework created by statistical approaches. The UNL Example Base is planned to serve for disambiguation purposes, once large analyzed corpora are available.

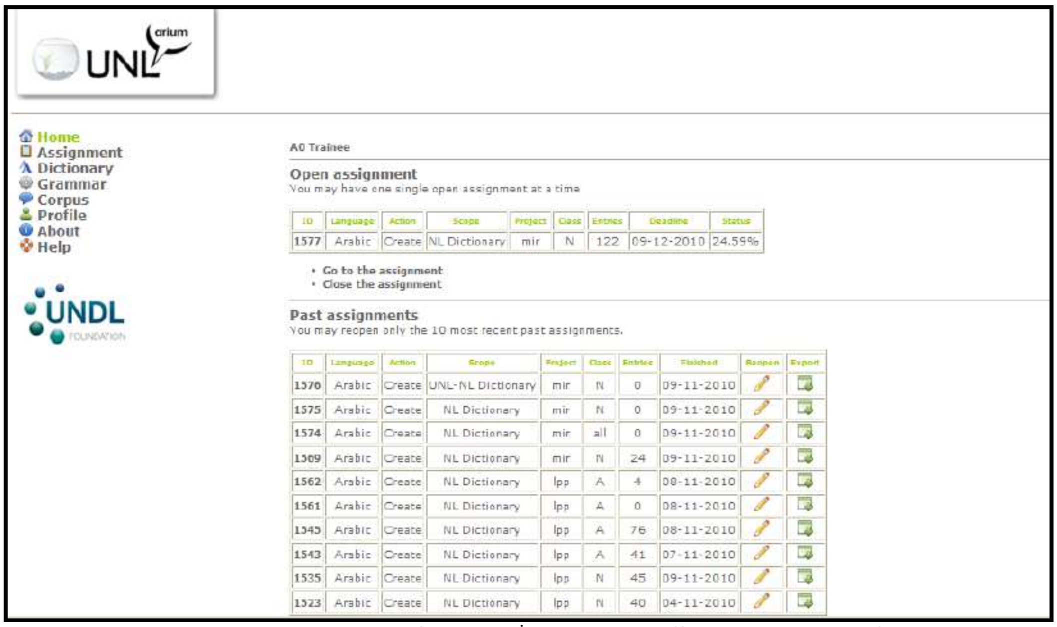
E. The UNL^{arium}

The UNL^{arium} is the integrated development environment used for producing all of the previous language resources (dictionaries, grammars, ontologies and corpora). It is a web-based collaborative database management system that allows registered users to create, edit and export language resources that have been created according to the UNL standards for language engineering. The UNL^{arium} also allows users to search, browse, and download dictionaries, grammars and corpora that have been provided by other users and in other languages.

Furthermore, the UNL^{arium} is considered a research workplace for exchanging information and testing the linguistic constants that have been proposed for describing and predicting natural language phenomena. Its main goal is helping the UNL system in devising a language-independent interlingua that would be as comprehensive and harmonized as required for NLP tasks. The UNL^{arium} comprises three main sections:

- Dictionary: for creating and editing dictionary entries;
- Grammar: for creating and editing inflectional paradigms, subcategorization frames, and other analysis and generation grammar rules; and
- Corpus: for adding, editing and exploring UNL documents.

Figure 18 shows the home page of the UNL^{arium} environment.



A0 Trainee

Open assignment

You may have one single open assignment at a time

ID	Language	Action	Scope	Project	Class	Entries	Deadline	Status
1577	Arabic	Create	NL Dictionary	mir	N	122	09-12-2010	24.59%

- Go to the assignment
- Close the assignment

Past assignments

You may reopen only the 10 most recent past assignments.

ID	Language	Action	Scope	Project	Class	Entries	Finished	Reopen	Export
1576	Arabic	Create	UNL-NL Dictionary	mir	N	0	09-11-2010	✎	⇩
1575	Arabic	Create	NL Dictionary	mir	N	0	09-11-2010	✎	⇩
1574	Arabic	Create	NL Dictionary	mir	all	0	09-11-2010	✎	⇩
1569	Arabic	Create	NL Dictionary	mir	N	24	09-11-2010	✎	⇩
1562	Arabic	Create	NL Dictionary	lpp	A	4	08-11-2010	✎	⇩
1561	Arabic	Create	NL Dictionary	lpp	A	0	08-11-2010	✎	⇩
1545	Arabic	Create	NL Dictionary	lpp	A	76	08-11-2010	✎	⇩
1543	Arabic	Create	NL Dictionary	lpp	A	41	07-11-2010	✎	⇩
1535	Arabic	Create	NL Dictionary	lpp	N	45	09-11-2010	✎	⇩
1523	Arabic	Create	NL Dictionary	lpp	N	40	04-11-2010	✎	⇩

Figure 15: The home page of the UNL^{arium} showing its different components (left)

The UNL^{arium} is free and open to any individual or institution wishing to participate in the furtherance of the UNL program. It intends to be as linguist-friendly as possible, and targets language specialists rather than computer experts. The UNL^{arium} does not require any thorough knowledge in UNL, NLP or Computational linguistics. It, however, requires that participants be of some acquaintance with descriptive Linguistics and have very good knowledge of the working language as well as English which is (for the time being) the language of the interface and of all the documentation. They also have to be certified in VALERIE, the Virtual Learning Environment of UNL (mentioned earlier in section 2). Non-certified users will, nevertheless, still have access to several facilities of the UNL^{arium}, but will not be allowed to add or edit entries or rules. Participants can work in providing the required language resources for the projects and the languages hosted by the UNDLF either as volunteers, or freelancers to be remunerated for their work, depending on the projects and on the languages. Being such a collaborative project, the data stored in the UNL^{arium} is available under an Attribution Share Alike (CC-BY-SA) Creative Commons license[5]. The UNL^{arium} can be accessed at (http://www.unlweb.net/unlarium/).

5 UNL ENGINES AND TOOLS

UNL engines and tools include all the non-linguistic components used in developing, editing and using the language resources that are part of the UNL system. They include the main analysis tools and generation tools, as well as other supporting tools.

A. Analysis Tools

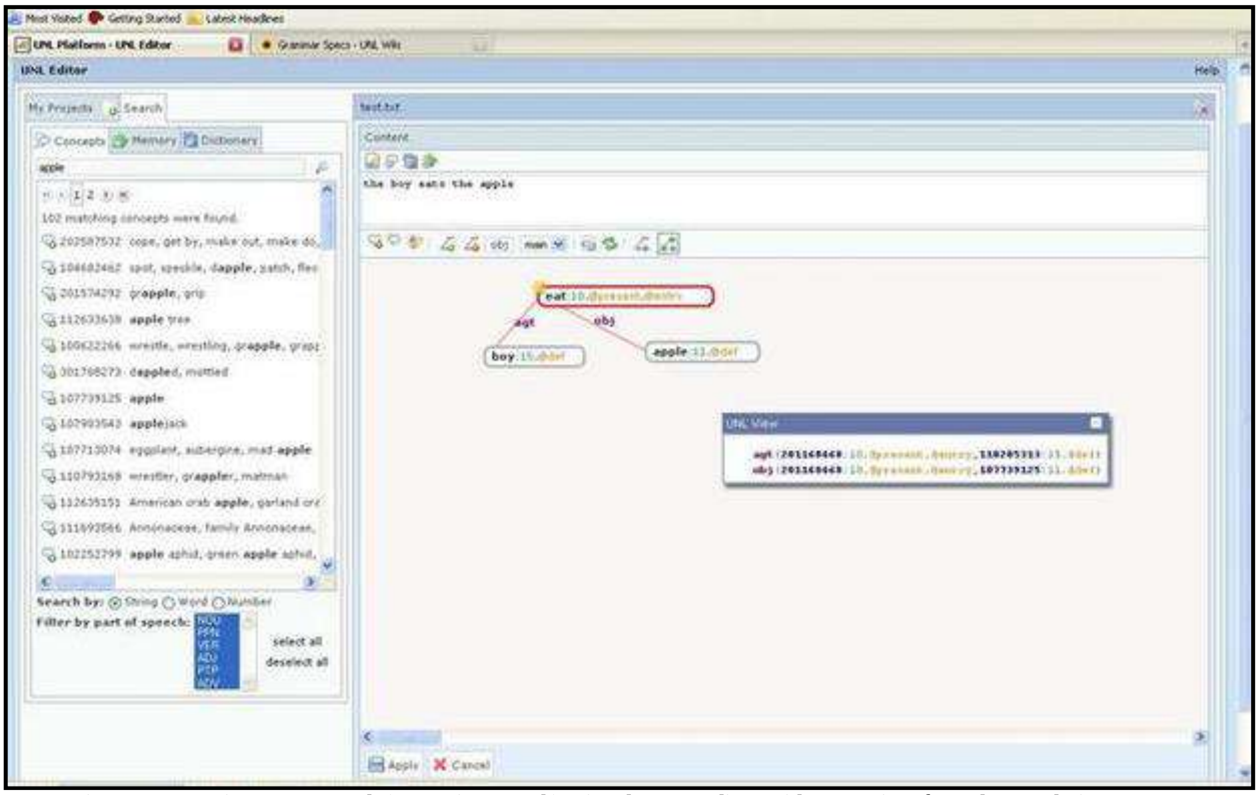
Of course, older UNL systems used analysis tools (Information about the older engines can be found at (http://www.undl.org/unlsys/ds.htm); however, these tools suffered from many shortcomings and were, therefore, replaced in the UNL^{+3} plan by more efficient ones. The tools used in analysis are the UNL editor, IAN, SEAN and the UNL memory base.

1) ***The UNL Editor:*** One of the main motives behind implementing the UNL^{+3} program is that the older analysis system was too inept to achieve the ambitious goals of UNL. The older system aimed for all analysis processes to take place in a semi-automatic manner, this proved to be too ambitious especially that the UNL system was still in its infancy. Thus, in the UNL^{+3} system, a more realistic manual analysis engine is introduced; namely, the UNL editor. The UNL editor is supposed to take over the analysis process until the disambiguation devices employed by the system mature enough for the semi-automatic engine (IAN) to operate. Currently, decision-making in the analysis process is mostly human and is performed using the UNL editor.

The UNL editor is a graph-based UNL authoring tool that facilitates the work of developers by providing them with a functionality to load/create UNL documents and manage their contents. The language specialist would upload the text to be

[5] http://creativecommons.org/licenses/by-sa/2.5/ch/

analyzed, select the appropriate UWs (i.e., the nodes in the semantic network) from the UNL dictionary, choose the semantic Relations that best represent the link between these nodes, and, finally, assign the nodes the required Attributes. The UNL editor is especially crucial for tasks demanding meticulous interpretation in order to yield accurate translations. The system has been recently improved with a base of successful past UNLizations to act as a translation memory (the UNL memory base to be discussed in subsection 4 of this section). Figure 19 shows the UNL editor in action.



```
apple
102 matching concepts were found.
202587532 cope, get by, make out, make do,
104682462 spot, speckle, dapple, patch, fleck
201574292 grapple, grip
112633638 apple tree
100622266 wrestle, wrestling, grapple, grapp
301768273 dappled, mottled
107739125 apple
107993543 applejack
107713074 eggplant, aubergine, mad apple
110793168 wrestler, grappler, matman
112635151 American crab apple, garland cra
111693566 Annonaceae, family Annonaceae,
102252799 apple aphid, green apple aphid,
```

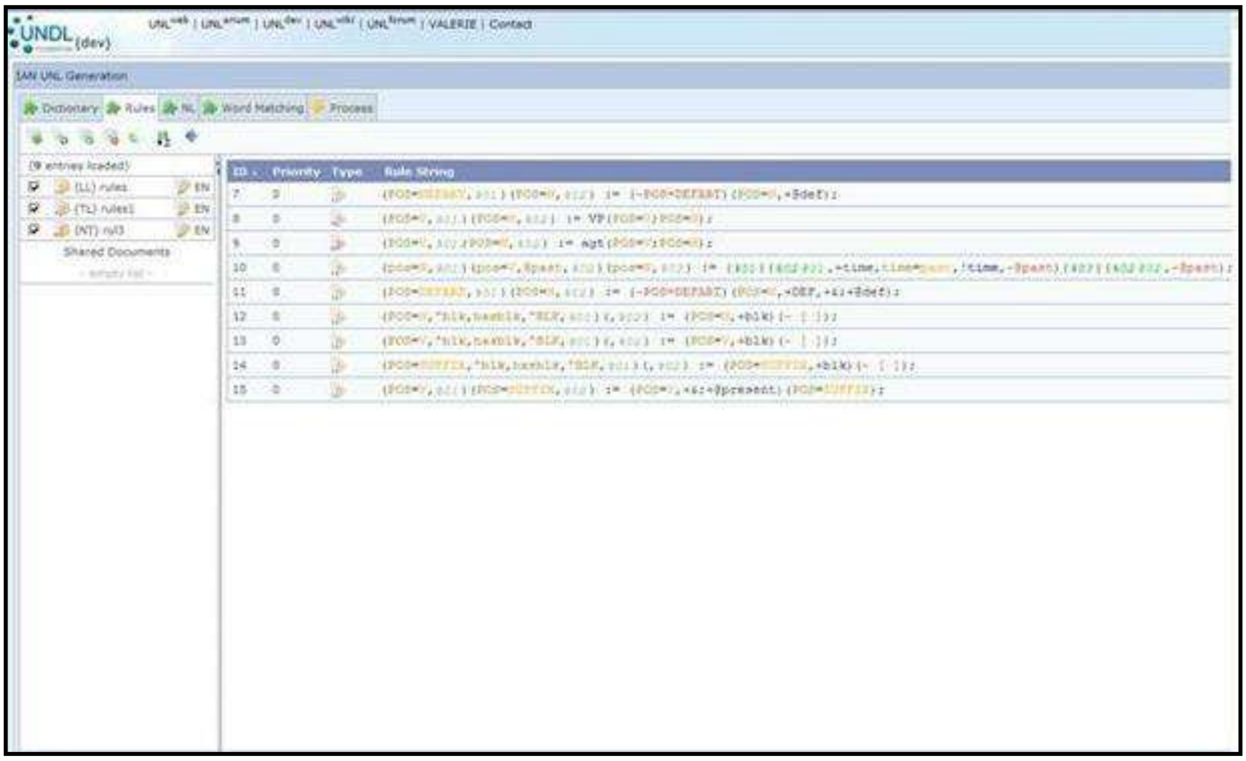
```
the boy eats the apple
```

```
agt(201168468:10.@present.@entry,110285313:15.@def)
obj(201168468:10.@present.@entry,107739125:11.@def)
```

Figure 16: Creating a UNL graph (semantic network) using the UNL editor, with suggestions from the translation memory

1) *IAN*: IAN is the semi-automatic analysis engine that replaced the older one (the Enconverter). The main shortcoming of the Enconverter was that it was designed with no respect to linguistic considerations. It did not employ any particular linguistic theory, and, as mentioned before, had no room for the different categories of linguistic issues; analysis was performed in a mostly automatic single leap. Inevitably, many problems arose; thus, the new engine, IAN, operated according to a more linguistic and realistic design. It employs five different modules (the five stages stated in the section 4 *B* 1 above) that process the input sentence step by step finally reaching the UNL semantic network.

IAN, the Interactive ANalyzer, is the natural language text analysis engine. It is the same for all languages, it simply employs the analysis grammar rules and the NL-UNL dictionary of the source language to analyze input and finally generate its corresponding UNL expressions. It operates semi-automatically; word sense disambiguation is still carried out by a language specialist, nevertheless, the system can filter the candidates using an optional set of disambiguation rules. Syntactic processing, on the other hand, is carried out automatically using the natural language analysis grammar, but syntactic ambiguities are referred back to the user, who may choose to backtrack to a different syntactic path. Nevertheless, human interaction is always optional, and is only used to improve the results. If no human intervention took place, the system would output the most likely alternative, which is the one with the highest priority in the lexicon and in the grammar. At any rate, IAN would be fully automatic as soon as the disambiguation devices it employs are perfected. Figure 20 shows a snapshot of the IAN engine in action.



ID	Priority	Type	Rule String
7	0		(POS=DEFART,%01)(POS=N,%02) := (-POS=DEFART)(POS=N,+$def);
8	0		(POS=V,%01)(POS=N,%02) := VP(POS=V)POS=N);
9	0		(POS=V,%01;POS=N,%02) := agt(POS=V;POS=N);
10	0		(pos=N,%01)(pos=V,$past,%02)(pos=N,%03) := (%01)(%02 [illegible],+time,time=past,!time,-$past)(%03)(%02 [illegible],-$past);
11	0		(POS=DEFART,%01)(POS=N,%02) := (-POS=DEFART)(POS=N,+DEF,+&:+$def);
12	0		(POS=N,^blk,haeblk,^BLK,%01)(,%02) := (POS=N,+blk)(- [ ]);
13	0		(POS=V,^blk,haeblk,^BLK,%01)(,%02) := (POS=V,+blk)(- [ ]);
14	0		(POS=SUFFIX,^blk,haeblk,^BLK,%01)(,%02) := (POS=SUFFIX,+blk)(- [ ]);
15	0		(POS=V,%01)(POS=SUFFIX,%02) := (POS=V,+&:+@present)(POS=SUFFIX);

Figure 17: The IAN engine using a set of UNlization rules.

3) SEAN: SEAN is the acronym for Shallow Enhanced ANalyser. It is a fully automatic analysis engine introduced in UNL^{+3}. In SEAN, the analysis targets the surface structure of natural language sentences; hence, SEAN is not appropriate for translation, but for information retrieval and extraction only since it only provides a rough and partial analysis of the natural language input. Unlike IAN, SEAN does not allow for any human intervention. Another difference is that SEAN does not allow a single document as an input (as in UNL editor and IAN), only a whole collection of documents. SEAN is also a word-driven analyser; the unit of analysis is the word, unlike IAN and EUGENE where the unit of analysis is the sentence.

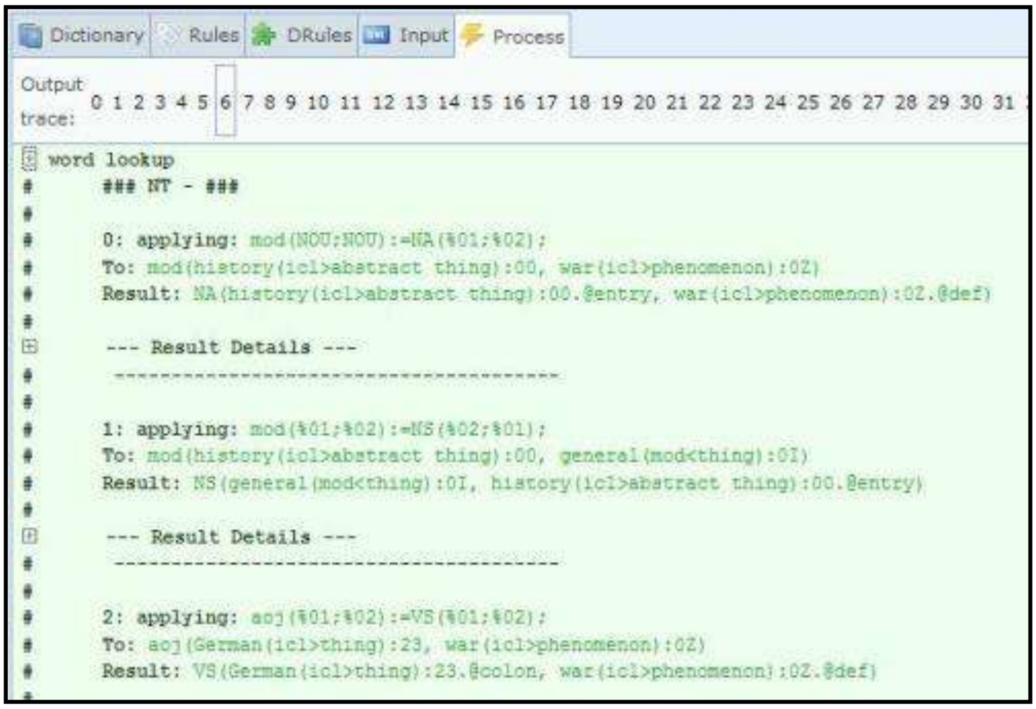
4) *UNL Memory Base*: The UNL Memory Base (UNL^{mb}) is a sort of a translation memory used to facilitate the analysis process by providing a repository of mappings between natural language structures and UNL graphs along with their frequency of occurrence. Mappings can be of continuous structures (such as n-grams) or discontinuous ones and can be extracted out of the UNL corpus or created by users. The UNL^{mb} can also be serve in disambiguating input texts. The UNL^{mb} will be populated by the results of the first project to be accomplished according to the specifications of UNL^{+3}; the iGLU project (to be discussed in section 8 *B*)

B. Generation Tools

Generation tools are those ones responsible for generating well-formed natural language sentences out of UNL semantic networks. Only one engine is used in the generation process; EUGENE.

1) EUGENE: The old generation engine, the DeConverter, has suffered from the same problem as the Encoverter, it did not take linguistic issues into account and assumed that generation should take place over a single step. Thus, similarly, it is replaced in the UNL^{+3} program by EUGENE (the dEep-to-sUrface natural language GENErator) that rather utilizes a five step generation process (the five stages mentioned earlier in section 4 *B* 2).

EUGENE, like the older engine, is a fully automatic engine; it does not involve any human intervention. Similar to IAN, EUGENE is language-independent, it simply uses the target language grammar rules and UNL-NL dictionary in order to decode the incoming UNL document and generate it in natural language format. Figure 21 shows a snapshot of EUGENE in action.



```
Output
trace:  0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31

word lookup
#       ### NT - ###
#
#       0: applying: mod(NOU;NOU):=NA(%01;%02);
#       To: mod(history(icl>abstract thing):00, war(icl>phenomenon):0Z)
#       Result: NA(history(icl>abstract thing):00.@entry, war(icl>phenomenon):0Z.@def)
#
        --- Result Details ---
#        --------------------------------------
#
#       1: applying: mod(%01;%02):=NS(%02;%01);
#       To: mod(history(icl>abstract thing):00, general(mod<thing):0I)
#       Result: NS(general(mod<thing):0I, history(icl>abstract thing):00.@entry)
#
        --- Result Details ---
#        --------------------------------------
#
#       2: applying: aoj(%01;%02):=VS(%01;%02);
#       To: aoj(German(icl>thing):23, war(icl>phenomenon):0Z)
#       Result: VS(German(icl>thing):23.@colon, war(icl>phenomenon):0Z.@def)
#
```

Figure 18: The Eugene engine applying generation rules

C. Other Supporting Tools

Apart from the main analysis and generation engines, several other tools are introduced by the UNL^{+3} plan to facilitate the handling of language resources and make better use of them.

1) Norma: a newly introduced tool is NORMA. Norma is a normalization tool developed to organize and consolidate raw knowledge bases. Normalization involves suppressing redundancies (relations with the same source and target nodes), suppressing tautologies (relations where the source node is the same as the target node), suppressing contradictions (opposite relations between the same nodes, or the same relations between opposite nodes), generalization (replacing a node by a hyper-node, or a relation by a hyper-relation), specification (replacing a hyper-node by a node or a set of nodes, or a hyper-relation by a relation), merging (replacing two nodes by a single node), and division (replacing one node by two or more nodes).

2) EDGES: EDGES is the Entity Discovery and Graph Exploration System, a user-friendly visualization tool used for exploring semantic networks by enabling node expansion, collapsing and navigation. It is expected to be embedded in several UNL-based applications discussed in section 6 of the paper.

D. The UNL^{dev}

In addition to enhancing the tools and engines, the UNL^{+3} program has also made the source codes for all tools open and their use free. The tools and engines are all available in the UNL Integrated Development Environment; the UNL^{dev}. The UNL^{dev} is a collective application containing the various UNL tools as well as the numerous applications to based on UNL infrastructure (some application will be discussed in section 8 of the paper). The UNL^{dev} can be reached at (http://dev.undlfoundation.org/index.jsp).

6 APPLICATIONS

This section examines some of the UNL-based front-end applications proposed in the UNL^{+3} plan. Applications are the main goal of developing the complex and intricate infrastructure discussed in all of the above. They should allow end users to perform several natural language processing tasks without any linguistic background or previous knowledge on UNL.

A. LILY

LILY is the acronym for Language-to-Interlanguage-to-Language sYstem, a UNL-based machine translation web service between the languages participant in UNL. LILY uses UNL as a pivot language; nonetheless, it has to be parameterized for each source and target language. Human intervention during the analysis process is allowed, however, it is optional. If no human intervention took place, the system chooses the results of the highest priorities in the dictionary and in the grammar. LILY can be assisted by the use of some of the UNL system's data banks such as the UNL^{eb}, UNL KB and UNL^{mb}. Thus, it may work either as a knowledge-based MT system or as an example-based MT system; however, it is always rule-based.

B. TUT

TUT (Text-to-Text through UNL) is a digital library of texts represented in UNL. It provides links to the original version of more than 30,000 titles and, the UNL version of the text (if available) along with three possible realizations (summarized, simplified and rephrased), in any of the languages available in the UNL System.

The main goal of TUT is to UNL-plicate texts. UNLplication involves generating several different versions of the same input, using UNL. These versions can be transformations of language (a version different in language), of length (text summarization, text extension), of structure (text, matrix, tree, graph) and of social adequacy (text simplification, colloqualization, sociolectalisation). UNLplication can extend the semantic formation of source documents to serve a wide range of uses that do not demand strict fidelity to the original. TUT is available at (http://www.unlweb.net/tut/).

C. KEYS

KEYS or the Knowledge Extraction sYStem is an information retrieval and extraction application that can search for information inside UNL documents. Thus, the search and extraction are language-independent and semantically-oriented yielding results that are more accurate than the results of the conventional string-matching systems. KEYS also includes the tools SEAN, NORMA and EDGES and is expected to synthesize and normalize the information available on the Web, and to provide summaries extracted out of several different input documents.

7 UNL WORLD

UNL World is the overall strategy followed by the UNDL foundation to propagate the UNL project, encourage the participation of institutions and individuals and disseminate knowledge about its scope, goals and applications. Projects involved in the category of UNL World include the UNL School which aims to train people who are interested in participating in the development of UNL-related modules, tools and applications. In accordance with the UNL School, the UNDL foundation has already organized several courses in Armenia, Switzerland, Egypt and India. Another component that can be considered part of the UNL World strategy is the UNL^{web} which comprises the UNL^{arium}, the UNL^{wiki} and the UNL forum.

8 CURRENT PROJECTS FOR B UILDING THE BASIC LINGUISTIC RESOURCES

After examining the five areas of interest composing the UNL^{+3} plan, this section will discuss some of the project that are underway according to the specifications of UNL^{+3}.

A. MIR

The UNL MIR is a collaborative project aiming at creating a general-purpose multilingual lexicon to be used in Natural Language Processing (NLP) tasks. The UNL MIR contains 27,255 entries representing different sets of synonyms (or synsets) of the English language automatically extracted out of an abridged version of the English WordNet 3.0. Each synset is also accompanied by a gloss and, occasionally, some examples to pinpoint its intended sense. The project, then, involves the different language centers associating each synset with the most appropriate lexical item in their respective languages and, hence, building their basic NL-UNL and UNL-NL dictionaries.

The UNL MIR provides a concept-to-word database (i.e., a semasiological, decoding or writer's dictionary) instead of a word-to-concept lexicon (onomosialogical, encoding, reader's dictionary); hence, entries in the UNL MIR should not be thought of as words, but as definitions to concepts along with their most likely lexical realizations in the language of the dictionary (synsets extracted from the WordNet represent the most likely lexical realizations of the concept in English).

The UNL MIR is open to the participation of individuals and institutions. The results of the project will be available under the Attribution Share Alike (CC-BY-SA) Creative Commons license mentioned earlier. More information about the UNL MIR project is available at (http://www.unlweb.net/unlweb/index.php?option=com_content&view=article&id=57:unl-mir&catid=1:latest-news&Itemid=60).

B. IGLU

The project i(GL>U) - from glosses into UNL - aims at UNLizing the glosses of 27,255 entries extracted from an abridged version of the WordNet 3.0. The results of the project will be used in populating the UNL KB; thus, improving the quality of word sense disambiguation and enhancing the capability of information retrieval and extraction through UNL. The results will also constitute the UNL memory base, to be used in future mappings between English and UNL.

The project is divided into two main phases. The first phase; (iGLU#1), addresses a subset of 27,255 synsets and will be carried out in a predominantly human basis. The second, on the other hand, will be devoted to the remaining 90,404 synsets and is expected to be mainly automatic. In the first phase, the working linguists would analyze (UNLize) WordNet glosses using the UNL Editor. Their decisions would then be stored in a UNL memory base, which would store the mappings between lexical items of English and Universal Words as well as the chosen attributes and relations. The data stored in the memory base will, then, be employed by IAN to automatically analyze the glosses of the second phase; a first step towards a fully-automatic natural language analysis system.

Similar to the UNL MIR, The iGL>U project is also open for the participation of individuals and institutions and the results of the project will also be accessible for all. More information about iGLU is available at (http://www.unlweb.net/unlweb/index.php?option=com_content&view=article&id=60:iglu&catid=1:latest-news&Itemid=60).

C. LACE

Up till now, UNL-based systems have been built upon lexical resources provided in a rather manual basis, mainly because the word sense disambiguation techniques has not yet reached the stage of making up for human intervention. Hence, the project LACE (Language Acquisition from Comparable tExts) aims at compiling, replicating and extending techniques that have been widely used in statistical natural language processing by making use of comparable corpora and building language modules out of data automatically extracted from comparable corpora; thus, extending the coverage of the current resources and providing a less expensive alternative for populating lexical databases in the UNL framework.

The results of the project would be used in natural language disambiguation, thus, improving the performance of the various UNL-based applications such as machine translation, summarization, information retrieval and semantic reasoning. More information about the project LACE is available at (http://www.unlweb.net/unlweb/index.php?option=com_content&view=article&id=66:lace&catid=1:latest-news&Itemid=60).

D. Le Petit Prince

In earlier phases, UNL technology had been used in UNLizing the text of *Le Petit Prince*, a French novel authored by Antoine de Saint-Exupéry in 1943. *Le Petit Prince* had been chosen as it is one of the best-selling books ever and has been translated to more than 180 languages and, thus, would serve as a good basis for contrasting and evaluating a wide range of UNL-based translations. Moreover, it provides new genres to experiment with; namely, narrative and literature.

The Project Le Petit Prince (LPP), hence, involves providing the UNL-NL dictionary entries and UNL-NL grammar rules required for automatically generating into natural language (NLizing) the already UNLized version of *Le Petit Prince*. The main goal is also to “UNL-plicate” the text in at least three different directions: replication, summarization and simplification, in as many languages as possible.

9 CONCLUSION

In less than three years, the UNL^{+3} program has succeeded in going a longer way than the previous 13 years of development have ever gone, thus, bringing the UNL system closer to its goals. However, this is understandable as such a massive project must require an extended period of trial and experimentation in order to detect and troubleshoot all the potential limitations and problems. UNL^{+3} specifications represent the compendium of these years, putting into effect the solutions to the detected problems and, by that, creating a UNL system that is more efficient, more easily managed and maintained and can output better results. Enhancements include visual and user-friendly environments, statistical tools in addition to the classic components of the UNL system, thus, bringing together a system that is almost ideal. By the end of 2011, UNL^{+3} should put an end to the period of UNL research and finally bring to light operational UNL-based applications that can serve the general public in performing fast and accurate natural language processing tasks. A compete presentation of how UNL^{+3} applies to the Arabic language will be presented in [13] in this volume.

REFERENCES

[1] Charles J. Fillmore, "The Case for Case" , In *Universals in Linguistic Theory*, Holt, Rinehart, and Winston. 1968.

[2] H. Uchida, 2006. Universal Networking Language (UNL): Specifications version 2005. Edition 2006. Available from: www.undl.org/unlsys/unl/unl2005-e2006/ (accessed 15th november 2010).

[3] H. Uchida, M. Zhu and T. Della Senta, "UNL: A gift for A millennium". Institute of Advanced Studies, United Nations University, Tokyo.1999.

[4] I. Boguslavsky, J. Cardeñosa and C. Gallardo. "A Novel Approach to Creating Disambiguated Multilingual Dictionaries", *Applied Linguistics*, vol. 30, 70–92, Oxford University Press, 2008.

[5] I. Boguslavsky, J. Cardeñosa, C. Gallardo and L. Iraola. "The UNL Initiative: An Overview*", Lecture Notes in Computer Science*, Volume 3406, 377– 87, Computational Linguistics and Intelligent Text Processing (CICLing), 2005, ISBN 978-3-540-24523-0.

[6] J. Bekios, I. Boguslavsky, J. Cardeñosa and C. Gallardo. "Using Wordnet for building an Interlingua Dictionary" in *proceedings of 5th International Conference on Information Reseasrch and Applications*, (I.TECH). vol.1, pp. 39-46, June, 2007.

[7] J. Cardeñosa, Alexander Gelbukh and Edmundo Tovar (eds.), *Universal Networking Language: advances in theory and applications*, Mexico City, National Polytechnic Institute. 2005.

[8] N. Chomsky. "Remarks on Nominalization" In: *Readings in English Transformational Grammar.* R. Jacobs and P. Rosenbaum (eds.), pp. 184-221. 1970.

[9] Noha Adly, Sameh Alansary, "Evaluation of Arabic Machine Translation System based on the Universal Networking Language", in *proceedings of 14th International Conference on Applications of Natural Language to Information Systems,* (NLDB 2009), Saarland University, Saarbrücken-Germany, June 24 - 26 2009.

[10] R. Martins, and V. Avetisyan, "Generative and Enumerative Lexicons in the UNL Framework", in p*roceedings of 7th International Conference on Computer Science and Information Technologies,* (CSIT 2009), 28 September - 2 October, 2009, Yerevan, Armenia. 2009.

[11] Ray S. Jackendoff, *X syntax: A study of phrase structure*, Cambridge, Massachusetts, MIT Press, 1977.

[12] Sameh Alansary, Magdy Nagi and Noha Adly,"Machine Translation Using the Universal Networking Language (UNL)", in *prooceedings of 8th International Conference on Language Engineering*, Ain Shams University, Egypt, December 18 – 19, 2008.

[13] Sameh Alansary, Magdy Nagi andNoha Adly, "A Practical Application of the UNL+3 Program on the Arabic Language", in *proceedings of 10th International Conference on Language Engineering*, Ain Shams University, Egypt, December 15 – 16, 2010.

[14] Sameh Alansary, Magdy Nagi, and Noha Adly, "A Semantic-Based Approach for Multilingual Translation of Massive Documents", in *proceedings of 7th Symposium of Natural Language Processing*, Thailand, December 13 - 15 2007.

[15] Sangharsh Boudhh, and Pushpak Bhattacharyya, "Unification of Universal Words Dictionaries using WordNet Ontology and Similarity Measures**",** in *proceedings of 7th International Conference on Computer Science and Information Technologies*, (CSIT 2009), Yerevan, Armenia, 28 September - 2 October, 2009.

[16] *WordNet: An electronic lexical database*. Christiane Fellbaum (Ed.), Cambridge, MA, MIT Press, pp. 423, 1998.

BIBLIOGRAPHY

Hiroshi Uchida. *UNL: Universal Networking Language – An Electronic Language for Communication, Understanding, and Collaboration*, United Nations University,Institute of Advanced Studies, UNL Center, Tokyo, Japan, 1996.

Hiroshi Uchidaand Meiying Zhu, *The Universal Networking Language beyond Machine Translation*, The UNDL Foundation, 2001.

Sameh Alansary, "Issues on Interlingua Machine Translation Systems", in *proceedings of 9th Conference on Language Engineering*, Cairo, December 23-24, 2009.

Sameh Alansary, Magdy Nagi and Noha Adly, "Generating Arabic Text: the Decoding Component in an Interlingual System for Man-Machine Communication in Natural Language", *in proceedings of 6th International Conference on Language Engineering*, Cairo, Egypt, December 6 - 7 2006

Sameh Alansary, Magdy Nagi, and Noha Adly, "Communicating in Arabic in Cyberspace", *in proceedings of Information and Communication Technology International Symposium*, (ICTIS07), Arabic Natural Language Processing Workshop, Fez, Morocco, April 3 - 5 2007.

Sameh Alansary, Magdy Nagi, and Noha Adly, "The Universal Networking Language in Action in English-Arabic Machine Translation", *in procedings of 9th Conference on Language Engineering*, Cairo, December 23-24, 2009.

Sameh Alansary, Magdy Nagi, and Noha Adly,"Processing Arabic Text Content: The Encoding Component in an Interlingual System for Man-Machine Communication in Natural Language", in *proceedings of 6th International Conference on Language Engineering*, Cairo, Egypt, December 6 - 8 2006.

Ontology-based Architecture for an Arabic Semantic Search Engine

Ibrahim Fathy Moawad*1, Mohammad Abdeen**2, Mostafa Mahmoud Aref**3

**Information System Department, Faculty of Computer Science and Information Sciences, Ain Shams University
Abbassia, Cairo, Egypt

2ibrahim_moawad@hotmail.com

*Computer Science Department, Faculty of Computer Science and Information Sciences, Ain Shams University
Abbassia, Cairo, Egypt

3m_abdeen2@yahoo.ca

3aref_99@yahoo.com

***Abstract** Most of the current Arabic search engines are classified as syntactic search engines, since the search is based on keyword(s). These search engines present several problems related to the meaning of the search query. For example, the low query precision and the shortness in understanding user's query intention represent some of these problems. In this paper, an Arabic semantic search engine based on an Arabic ontology. The proposed architecture is layered, and is loosely coupled with an existing Arabic syntactic search engine. The proposed Arabic semantic search engine is semantically reason using an Arabic ontology that represents a very rich vocabulary (Arabic concepts' attributes, inheritance relations, and association relations). It helps the search engine to understand the user's query intention, and hence enhances the search results. Finally, the paper illustrates semantic search through simple search examples in computer domain.*

1. INTRODUCTION

The internet has become the world's biggest information superstore. From simple bloggers to space agencies, all use the internet fabric for information sharing. The size of Internet is doubling every 5 years. Figure 1 shows the number of Internet sites from 1995 to 2008. Users locate their needed information through search engines [1].

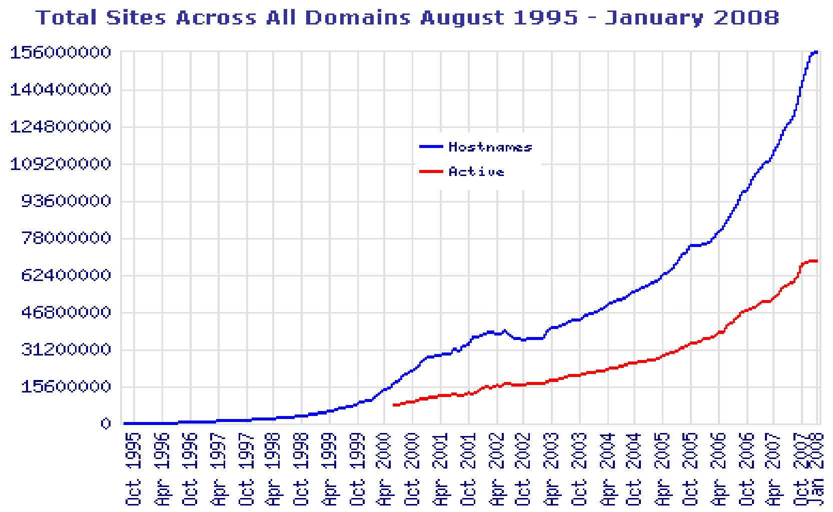


Figure 1: The number of Internet sites from 1995 to 2008.

Existing search engine technology works well in a narrow set of situations. Such as when the user is able to provide search terms that precisely match the resources they are attempting to locate. As the number of resources that can be accessed and searched by computer users increases, search engines are no longer just convenient information tools. In fact, they are powerful agents of a transformation that is making the business environment more transparent, and thus, potentially more competitive. While the development of search engines has significantly increased the ability of computer users to discover or locate information, existing search engine technology still has various significant limitations, and it is frequently insufficient to help people locate the information they need.

The alternative of keyword-based search engines is having semantic search engines that generate information more relevant to the user need [2]. The semantic search engines use ontology rather than lexicon that is used in the traditional search engines.

An Arabic ontology is needed to achieve the Arabic semantic search engine. In this paper, the Arabic ontology is going to be used to enhance an existing Arabic search engine by developing a semantic Arabic search engine.
Section 2 presents the background and the related work about semantic search engines and Arabic Ontology. Section 3 discusses the architecture of the proposed Arabic semantic search engine. Arabic semantic search examples in computer domain are illustrated in section 4. Conclusion and future works are given in section 5.

2. BACKGROUND

Semantic search is a process used to improve online searching by using data from semantic networks to disambiguate queries and web text in order to generate results that are more relevant. A Semantic network (also called "concept network" or ontology) is a graph, where vertices represent concepts and edges represent relations between concepts. At the level of ontology, a semantic network expresses vocabulary that is helpful especially for human, and can be used for machine processing.
A Semantic Search Engine attempts to make sense of search results based on context. It automatically identifies the concepts structuring the texts [3]. For instance, if you search for “election”, a semantic search engine might retrieve documents containing the words “vote”, “campaigning” and “ballot”, even if the word “election” is not found in the source documents. An important part of this process is disambiguation of both the user queries and the web content. This means that the search engine, through natural language processing, will know whether you are looking for a car or a big cat when you search for “jaguar” [4].

A. The Semantic Search Engines

In recent years, considerable research efforts have been devoted to apply semantic web technologies into information search and retrieval process. There are a number of pilot non-Arabic semantic search projects and frameworks have been implemented and evaluated in various application domains. Some of these semantic search engines can be summarized as follow: -

Hakia [5] is a general purpose semantic search engine, as opposed. It searches structured corpora (text) like Wikipedia. Often, Hakia will propose related queries, which is also great for research. For instance, if we search for Barack Obama, Hakia suggest that we might be interested in information about Michelle Obama, Hillary Clinton, Democrats, Sarah Palin, John McCain, John Sununu and Joseph R. Biden Jr. as well.

SenseBot [6] is a web search engine that summarizes search results into one concise digest on the topic of your query. The search engine attempts to understand what the result pages are about. For this purpose, it uses text mining to analyze Web pages and identify their key semantic concepts.

Cognition [7] has a search business based on a semantic map, built over the past 24 years, which the company claims is the most comprehensive and complete map of the English language available today. It is used in support of business analytics, machine translation, document search, context search, and much more. You can use Cognition’s technology to search one of four bodies of information: public (Resource.org is currently contains 1,858 volumes consisting of 675,704 files of federal case law in XHTML format), MEDLINE (Medical Literature Analysis and Retrieval System Online) abstracts, the English version of Wikipedia, and the complete New English Translation including text and translator notes of the Gospels of Matthew, Luke, John and Mark.

B. Arabic Ontology

Ontology is defined as an explicit specification of conceptualization. Ontology will thus analyze the most general and abstract concepts or distinctions that underlay every more specific description of any phenomenon in the world, e.g. time, space, matter, process, cause and effect, system [8]. There are several works done in English ontology. The most common used ones are Upper Model [9], Wordnet [10], Sumo [11] and OpenCyc [12]. For the Arabic language, little work has been done on the subject of Arabic ontology. The most common work is Arabic Wordnet [13, 14].
In general, there are two main kinds of Ontology [15]:

1. **A domain ontology (or domain-specific ontology)** that models a specific domain, or part of the world. It represents the particular meanings of terms as they apply to that domain. For example the word "card" has many different meanings. An ontology about the domain of poker would model the "playing card" meaning of the word, while an ontology about the domain of computer hardware would model the "punched card" and "video card" meanings.
2. **An upper ontology (or foundation ontology)** is a model of the common objects that are generally applicable across a wide range of domain ontologies. It contains a core glossary in whose terms objects in a set of domains can be described. There are several standardized upper ontologies available for use, including Dublin Core, GFO, OpenCyc/ResearchCyc, SUMO, and DOLCE. OntoSelect [16] is a library that monitors the webs that provide an access point for ontologies on any possible topic or domain that is automatically updated, organized in a meaningful way and with support for ontology search and selection. Based on OntoSelect, figure 2 shows the distribution of languages used in creating Ontologies on the

Semantic Web. We can detect from the graph that the Arabic Language isn't considered, due to the lack of a real considerable and reliable Arabic Ontology.

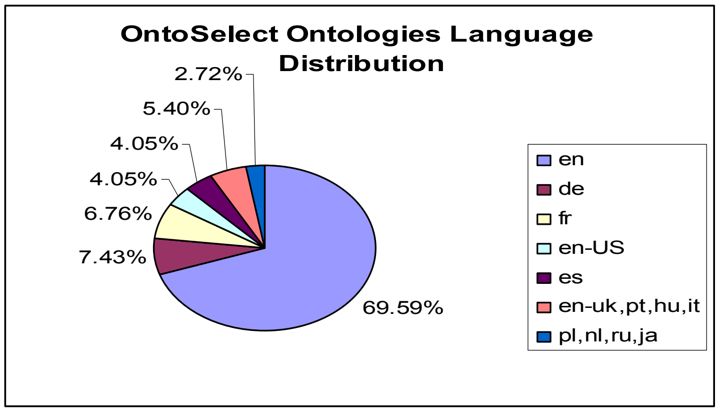


Figure 2: OntoSelect: Distribution of languages used in creating ontologies

3. THE ARCHITECTURE OF ARABIC SEMANTIC SEARCH ENGINE

The current Arabic keyword-based search engines present several problems related to the meaning of the keyword used in the search query [17]. In order to solve the problems of the low query precision and the shortness in understanding user's query intention that occur in these traditional search engines, this paper aims to propose an Arabic search engine architecture based on Arabic ontology. By using semantic reasoning, which based on ontology, it helps the search engine to understand the user's query intention, and hence enhances the search results.

The proposed Arabic search engine architecture will be developed as a loosely coupled semantic layer over an Arabic syntactic search engine. Building an Arabic semantic search engine will consider related semantic Arabic words when searching for an Arabic one (Arabic Ontology). The Arabic semantic search engine (shown in Figure 3) will utilize the existence of Syntactical Search Engine. Two modules are going to be added to a Syntactic Search Engine: Semantic Query Analyzer and Semantic Ranker.

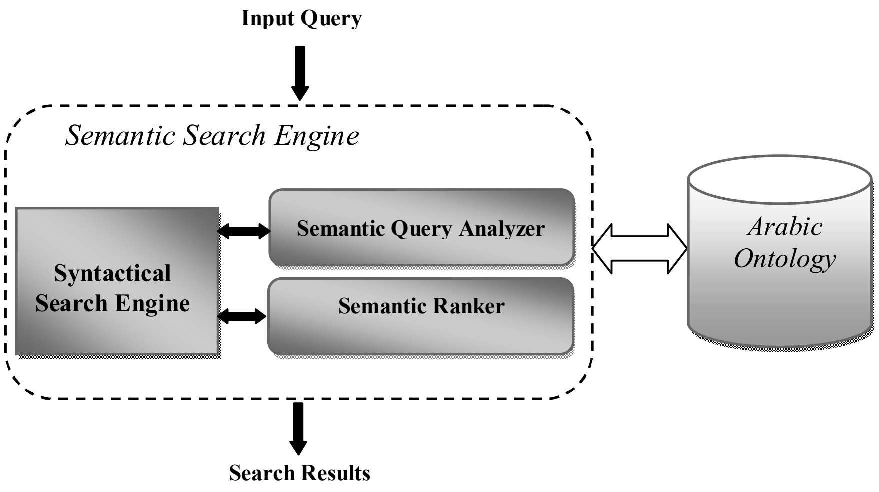


Figure 3: The Ontology-based Arabic Search Engine Architecture

As shown in Figure 3, the ontology-based Arabic search engine consists of an Arabic Ontology module, a semantic query analyzer module, a semantic ranker and a syntactic search engine. To develop the architecture shown in figure 3, there are four main objectives should be achieved: Building an Arabic Ontology using limited vocabulary; Developing a Semantic Query Analyzer module, Developing a Semantic Ranker, and Interfacing with the existed Syntactical Search Engine. A brief description of these objectives is given as follows:

1. **Building an Arabic Ontology**
   There are two approaches to build ontology: top-down and bottom-up [18]. In the top-down approach, the top of the hierarchy will be considered first (e.g. entity). The second level will be considered next (abstract entity or concrete entity), and so on. On the other hand, in the bottom-up approach, concepts are grouped and analyzed. The attributes

(properties) of these concepts are specified. The common attributes of group of concepts are assigned to a higher level concept. The group concepts will be considered as children for this higher concept. Constructing ontology for the whole Arabic Language is a very difficult task. In this paper, we consider a subset of the Arabic language. This subset might be domain specific or limited vocabulary. The limited vocabulary is considered in this work. The limited vocabulary is then used to build an Arabic semantic search engine.

2. **Developing the Semantic Query Analyzer**
   Given the user query, a semantic query analyzer accesses the Arabic ontology to find the related concepts (semantically related words). These concepts are sent to the syntactic search engine to find the related documents in its indexed database.
3. **Developing the Semantic Ranker**
   In syntactic search engines, search results areranked before delivered to the query generator (the user). Usually the "relevant" documents generated by the search query are numerous (could be in hundreds of thousands or even more). Displaying these documents to the user at one time and with no intervention or prioritization could yield the search useless. Users are usually interested in the first 10 documents or so and will not bother looking at the remaining ones. It is therefore of importance to present the search result to the user in the most effective way by showing the most relevant documents first. This is known as "page ranking".
   While in syntactic search engines the page rank is calculated using the "term frequency", this same technique is not appropriate for semantic search engines since the produced documents could have many incidences of the concept without having the same term as in the search query. For semantic search engines, possible words of the same concept will have to be generated with the help of the Ontology. The result will be used by the semantic ranking module to perform the ranking.
4. **Interfacing with Syntactical Search Engine**
   To utilize the existing syntactical search engine, a set of application interfaces should be defined and developed. These interfaces are used to loosely couple the semantic ranker module and the semantic query analyzer module with the syntactical Search Engine.

4. ARABIC SEMANTIC SEARCH EXAMPLES IN COMPUTER DOMAIN

To validate the proposed Arabic semantic search engine architecture, set of simple examples in computer domain have been experimented. The developed prototype helps the end user to compose the search query semantically, where it includes a module called "**Interactive Semantic Query Analyzer**". This module interacts with the end user by recommending him with extra semantic search criteria by accessing specific-domain ontology (a very simple ontology in the **Computer domain**). Also, the developed prototype exploits the Google search engine by accessing it using the **Google APIs**. Figure 4 shows the prototype architecture that includes three components: Interactive Semantic Query Analyzer, Computer Specific Ontology, and the Google search engine. Interactive Semantic Query Analyzer has two main roles: interacting with the end user to build search query based on the ontology semantics, and interfacing with the Google by sending the composed search query. After that, the search results are retrieved and appeared to the user by Google. Three test cases have been experimented.

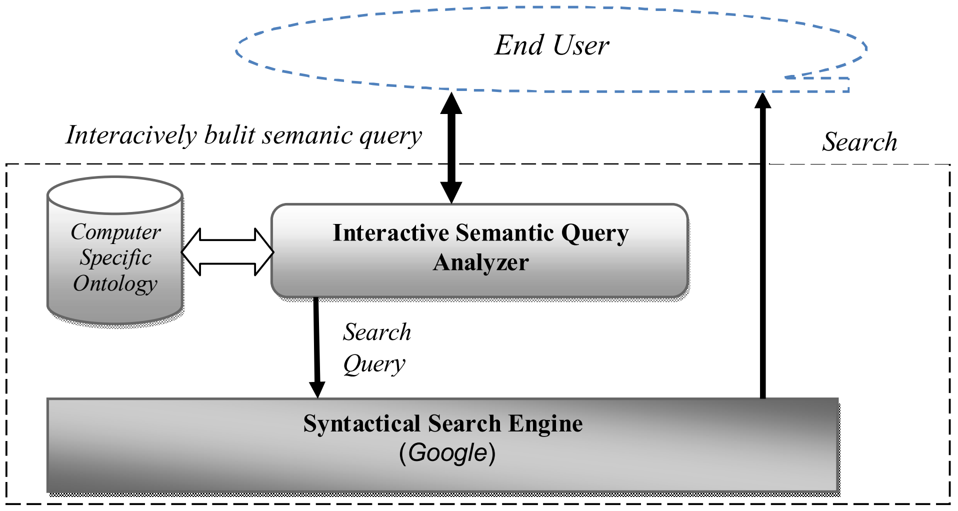

End User
Interacively bulit semanic query
Search
Computer Specific Ontology
Interactive Semantic Query Analyzer
Search Query
Syntactical Search Engine (Google)

Figure 4: Experimental Prototype Architecture

A. *ONTOLOGY OVERVIEW*

Figure 5 shows the computer specific domain ontology used in the developed prototype. This ontology contains set of concepts like "الحاسب" , "العتاد" , etc. Also it contains three types of relations: -

1. **Inheritance relations**: Both the "**أقراص ممغنطة**" concept and the "**أقراص ضوئية**" concept inherit the "**العتاد**" concept. Also, the "**إنشاء الوثائق**" concept, the "**حل المسائل الرقمية**" concept, and the "**تخزين واسترجاع المعلومات**" concept inherit the "**مجال**" concept.
2. **Association relations**: The "**يطبق فى**" relation relates both the "**الحاسب**" concept and the "**مجال**" concept. Also, The "**يتبع**" relation relates both the "**الحاسب**" concept and both the "**برامج**" concept and the "**العتاد**" concept.
3. **Synonym relations**: Both "**الحاسب**" concept and "**الكمبيوتر**" concept are synonyms. Also, Both "**برامج**" concept and "**تطبيقات**" concept are synonyms.

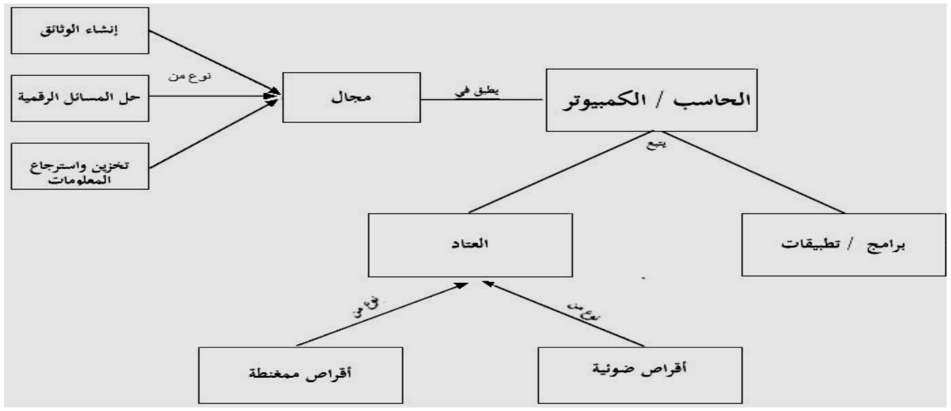


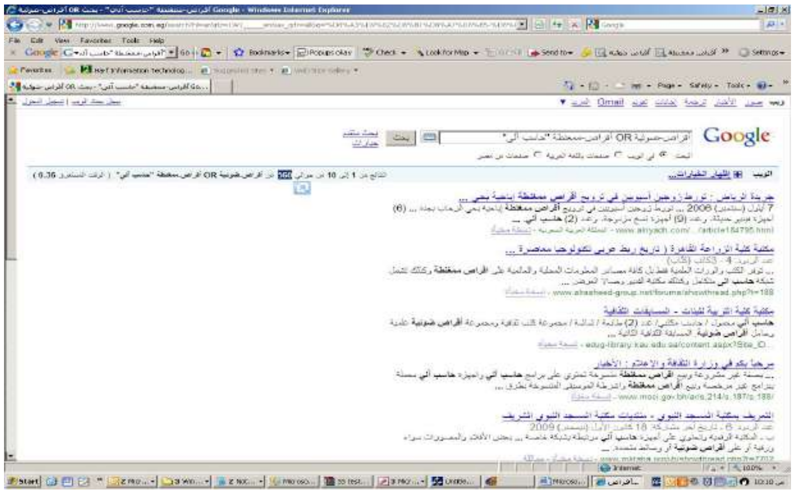
Figure 5: Computer domain specific ontology

B. *Test Cases*

The following are two examples that have been experminted using the system prototype shown in figure 4, and the computer domain ontology shown in figure 5. The following two cases are illustrated in terms of a brief description of each case including its interactive search scenario, both syntactic search (using Google only) and the semantic search results, and finally a brief comparison between both results.

1) **Case 1:** Inheritance relation search

Let's assume that the user would like to perform a query such as "أقراص ضوئية حاسب آلي ", the Interactive Semantic Query Analyzer module suggests that it is better to search also with " أقراص ممغنطة ", where both "أقراص ضوئية حاسب آلي " and " أقراص ممغنطة" concepts are sub-types of another concept called "عتاد ". Figure 6 shows the semantic search results of case 1 (560 pages), where figure 7 shows the syntactic search results (12900 pages). As noted, the number of semanitc search results is very small if it is compared with the number of syntactic search results (it is 4.3% of the traditional search results).



أقراص-ضوئية OR أقراص-ممغنطة "حاسب آلي"

النتائج من 1 إلى 10 من حوالي 560 عن أقراص-ضوئية OR أقراص-ممغنطة "حاسب آلي" (الوقت المستغرق 0.36)

[illegible]

Figure 6: The semantic search results of case 1

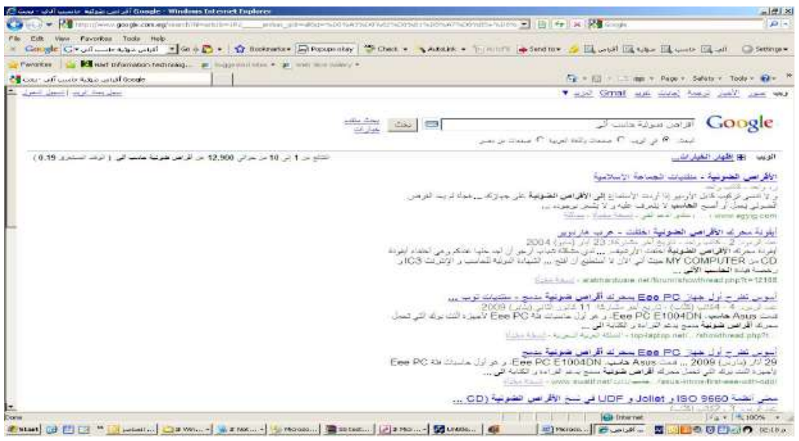


Figure 7: The syntactic search results of case 1

2) **Case 2:** Association relation search

In this case, let's assume that the user would like to perform a query such as "تطبيقات الحاسوب ", the Interactive Semantic Query Analyzer module suggests that you can also search with associated concepts with "تطبيقات الحاسوب ", which are "حل المسائل الرقمية" "تخزين واسترجاع المعلومات" "إنشاء الوثائق والصور". Figure 8 shows the semanitc search results of case 1 (64200 pages), where figure 9 shows the syntactic search results (62900 pages). As noted, both number of semanitc search results and syntactic search results are approximatly similar, but the semantic search results include more specfic and user-accepted query keywords.

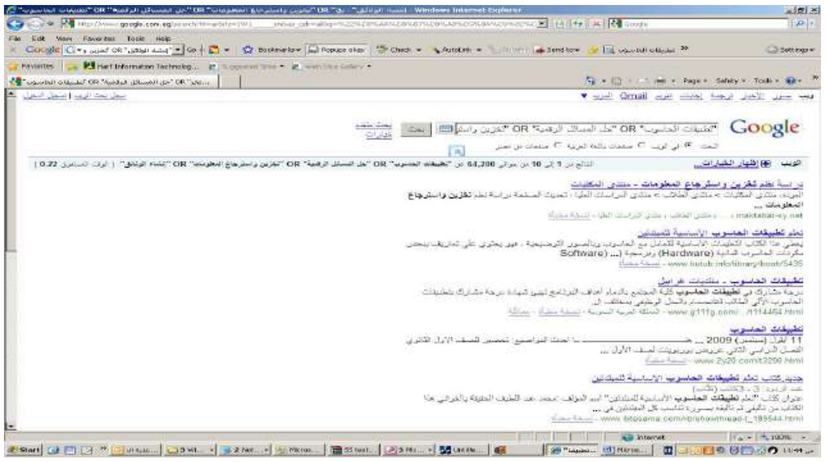


Figure 8: The semantic search results of case 2

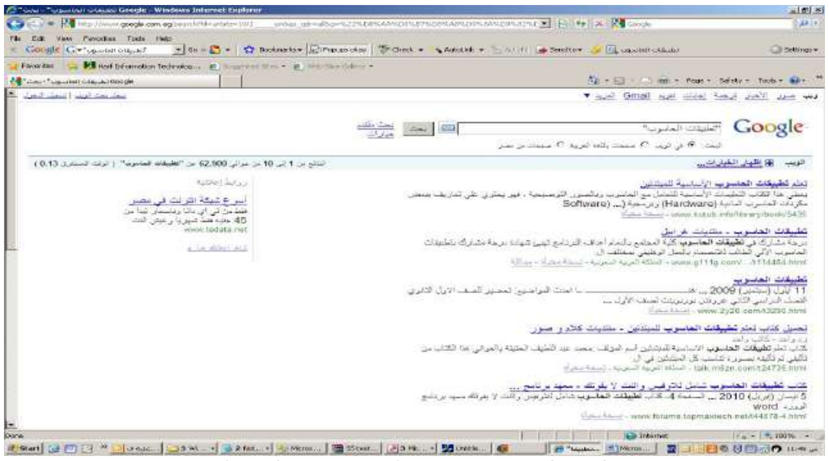


Figure 9: The syntactic search results of case 2

5. Conclusion

The majority of the currently available search engines are keyword-based. While these engines are relatively easy to use, their search results in some cases could be inaccurate and sometimes misleading. Semantic search engines present a more viable and less ambiguous alternative to the keyword-based search engines. Several works are found in the literature that proposes semantic search engines based on ontology. In those works the supported ontologies and therefore search engines are for the English language. In this paper we have proposed a semantic search engine for the Arabic language. This engine is based on an Arabic ontology. We have adopted the limited vocabulary Arabic ontology approach. We have shown an overall architecture for the proposed Arabic semantic search engine. This architecture consists of three main modules: the semantic query analyzer and the semantic page ranker modules. The third module is the regular keyword-based search engine.We have demonstrated the effectiveness of this architecture with two examples. Both examples are based on the computer domain. We have also compared the results from our semantic search engine with that of a syntactic search engine (Google). Our results showed that the number of pages using semantic query are far less than that of a syntactic search. This helps a great deal with more accurate search results and to reduce any search ambiguity.

References

[1] http://news.netcraft.com/archives/2009/01/16/january_2009_web_server_survey.html

[2] R. Guha, R. McCool, and E. Miller, “Semantic search,” in Proc. of the 12th international conference on World Wide Web, New Orleans, 2003, pp. 700–709.

[3] WWei, P M Barnaghi and A Bargiela, “The Anatomy and Design of A Semantic Search Engine”, Tech. rep., School of Computer Science, University of Nottingham Malaysia Campus, 2007.

[4] http://www.pandia.com/sew/1262-top-5-semantic-search-engines.html

[5] http://www.hakia.com/

[6] http://www.sensebot.net/

[7] http://www.cognition.com/

[8] Marek Obitko (advisor Vladimir Marik): Translations between Ontologies in Multi-Agent Systems, Ph.D. dissertation, Faculty of Electrical Engineering, Czech Technical University in Prague, 2007.

[9] John Bateman & Till Mossakowski, “Ontologies for spatial reasoning, action and interaction”, University of Bremen, NIST Discussion. March 2006.

[10] Amaro, R., R. P. Chaves, P. Marrafa, and S. Mendes ``Enriching Wordnets with new Relations and with Event and Argument Structures'' In: Seventh International Conference on Intelligent Text Processing and Computational Linguistics , pp. 28 - 40, Mexico City, Lecture Notes in Computer Science, Springer-Verlag. 2006.

[11] Chow, I.C and Webster, J.J. Integration of Linguistic Resources for Verb Classification: FrameNet Frame, WordNet Verb and SUMO. Alexander Gelbukh (Ed.) LNCS 4394/2007 pp.1-11. Computational Linguistics and Intelligent Text (CICLing'07), Mexico City, Mexico. 2007.

[12] Stephen L. Reed and Douglas B. Lenat, “Mapping Ontologies into Cyc”, American Association for Artificial Intelligence, 2002.

[13] Black, W., Elkateb, S., Rodriguez, H, Alkhalifa, M., Vossen, P., Pease, A. and Fellbaum, C., “Introducing the Arabic WordNet Project”, in Proceedings of the Third International WordNet Conference, Sojka, Choi, Fellbaum and Vossen eds. 2006.

[14] Black, W., Elkateb, S., Rodriguez, H., Alkhalifa, M., Vossen, P., Pease, A. and Fellbaum, C., Introducing the Arabic WordNet project, Proceedings of the 3rd Global Wordnet Conference, Jeju Island, Korea,South Jeju, January 22-26, 2006.

[15] K. S. Esmaili, H. Abolhassani, “A Categorization Scheme for Semantic Web Search Engines” Proceeding AICCSA '06 Proceedings of the IEEE International Conference on Computer Systems and Applications, 2006

[16] OntoSelect Library http://olp.dfki.de/ontoselect.

[17] Al-Khalifa, H., Al-Wabil, A. (2007). The Arabic Language and the Semantic Web: Challenges and Opportunities. International Symposium on Computers and the Arabic Language. November 2007, Riyadh, Saudi Arabia.

[18] Gruber, T. R., "Toward Principles for the Design of Ontologies Used for Knowledge Sharing". In: International Journal Human-Computer Studies, 43(5-6):907-928, 1995.

Automatic Speech Segmentation Using Genetic Algorithm Based on Best Tree Encoding

Amr M. Gody*

*Electrical Engineering Department, Faculty of Engineering, Fayoum University, Fayoum EGYPT

amg00@fayoum.edu.eg

Abstract ***Best Tree Encoding BTE features vector which was derived in a previous paper [1] is used to find phoneme boundaries along speech utterance. No training process is needed. Disturbance function is calculated for the given speech signal. Genetic algorithm is utilized to optimize the disturbance function for highlighting the phoneme boundaries. The given procedure indicates very promising results for speech recognition. It can trace the very hard transitions in phoneme stream along speech utterance. We can define the wrong markers as those exist in the result without a reference or those exist in the reference and do not exist in the result. Excluding the wrong markers, this procedure gives less than 10% drift in markers off the reference markers.***

1. Introduction

Speech signal is a stream of information. The basic information unit is called phoneme. It is believed that human speech is decomposed of small time durable unites called phonemes. Each phoneme contributes in specific piece of information. We can make analogy between spoken and written langue in such that phoneme is the basic unit in the spoken langue as the character is the basic unit in written langue. Information in each phoneme is encoded into the frequency domain. Simply the information is a pattern of frequency components [1]. Features are extracted from the speech signal to best represent such information. The good feature is ranked as of how far is it discriminating such information stream in speech signal.

Presently, manual annotation by expert phoneticians is the most precise way for time-aligning a speech waveform against the corresponding phonetic sequence. This is a tedious and time consuming task, which makes it a prohibitive choice for large speech corpora. Several approaches have been proposed for the task of speech segmentation [3-7]. The most frequently used approach is based on HMM phone models. In this method each speech waveform is initially decomposed into a sequence of feature vectors, using a speech parameterization technique. Afterwards, a set of HMM phone models (phone recognizer) is utilized to extract the corresponding phonetic sequence as well as the positions of the phonetic boundaries. Other speech segmentation methods have also been proposed in the literature. Some of them include detection of variations/similarities in spectral or prosodic parameters of speech, template matching using dynamic programming and/or synthetic speech and discriminative learning segmentation. Phone transitions (boundaries) may be classified into many types. It may be called phone transition type to refer to the transition between the left context phonetic class of a boundary to the right context class. The class may be classified as vowels, affricates, fricatives, nasals, glides, stops and silence.

Various speech parameterizations have been utilized in the phonetic segmentation task, with the Mel Frequency Cepstral Coefficients (MFCC) among the most widely used, especially in the HMM-based approach. Other speech features such as Perceptual Linear Prediction (PLP), Line Spectral Frequencies (LSF), Linear Predictive Coding (LPC), short-time energy, formants and wavelet-based have also been used.

In this research a new proposed speech parameter called Best Tree Encoding (BTE) is utilized for this task. BTE is first introduced in [1] and [2] as a new speech parameters. An approach that is much close to dynamic programming is utilized to solve this problem. The genetic algorithm is used to minimize the disturbance measure function for finding phone transitions. Considering that speech signal is stationary along the duration of phoneme utterance is the key point to find the boundaries. As it is supposed that speech is stationary along the phoneme duration, this is implies that minimum disturbance occurs in the extracted features along the phoneme utterance duration. Suitable function is designed to model the disturbance. Genetic algorithm is used to optimize the function parameters on a given signal under test. Speech database is prepared to measure the quality of this experiment. Speech database is labeled and transcribed then verified to evaluate the results of automatic segmentation. The following sections will navigate through the details of this research. Section 2 illustrates the problem definition. BTE speech feature is described in section 3. The disturbance measure function will be discussed in section 4. The derivation of the function will be clarified. In section 5, the Genetic algorithm will be illustrated to show how it is used to get phone transitions locations by minimizing the disturbance function. The results will be presented in section 6. Some examples that give the obtained phone locations and baseline speech signal marker locations will be introduced in this section. The conclusion will be given in section 7.

2. Problem definition

Accuracy in automatic Speech Segmentation is targeted in this research. Phoneme is the basic speech unit; it is supposed that there is some sort of homogeneity in a single phoneme. Figure 1 focuses on a segment of Arabic utterance that contains a transition between two phones. As shown in figure the properties in the frequency domain, the lower part of the figure, are almost homogeneous along each phone's duration. There is a transition region in which the properties are changing in time.

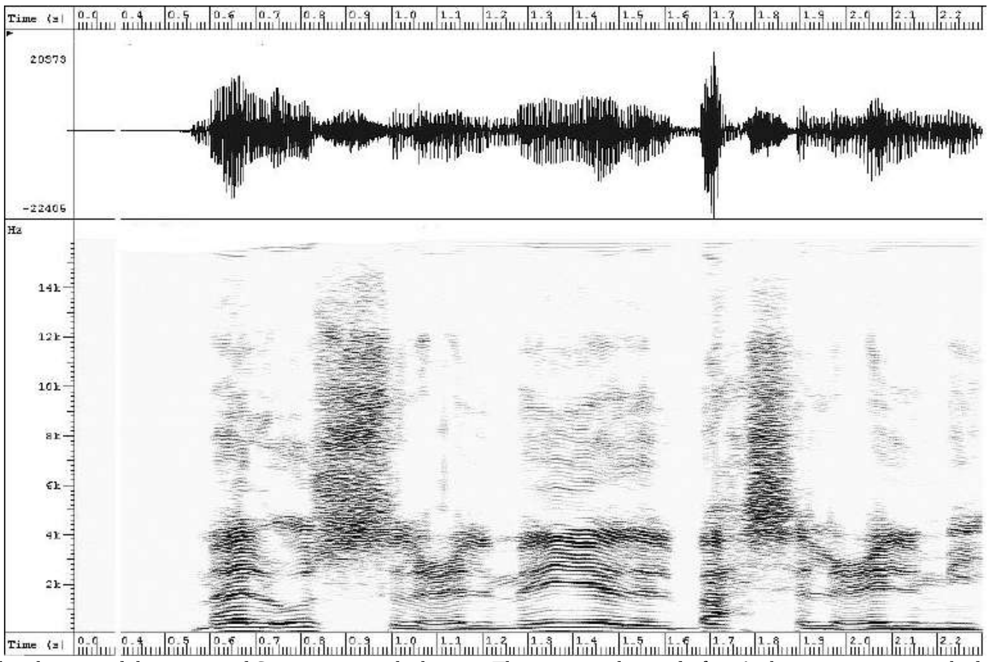


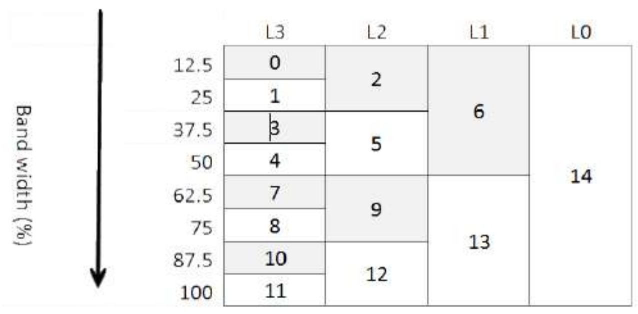
Figure 1 Speech signal at the top and the associated Spectrogram at the bottom. This is a speech signal of an Arabic utterance transcribed in Arabic letters as "مَاَ اَلْأُسُسِ اَلْعِلْمِيَّه لِلْبَحْثِ اَلْعِلْمِي". Spectrogram illustrates how phone transitions appear in frequency domain (the Y axes in the bottom figure)

It is required to find a model for describing phoneme homogeneity property in speech signal as illustrated in figure 1. It is also needed to find a disturbance measure of the model parameters. It is supposed that the disturbance will be a maximum at the phone boundaries.

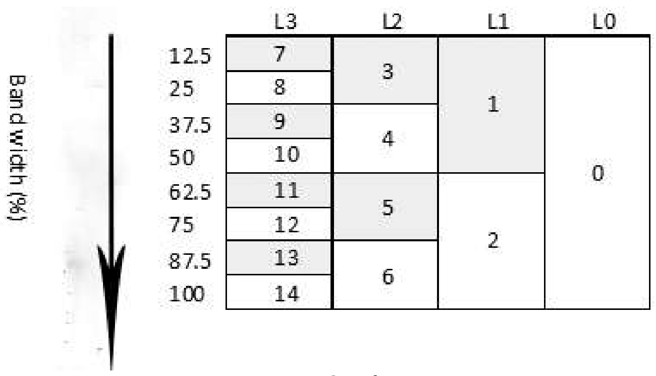
The model should reflect the frequency domain homogeneity that is shown in the lower part of figure 1. Best Tree Encoding (BTE) [1] is chosen as a map for speech signal in the frequency domain. Section 3 will give a summary of BTE model and its suitability for this point of research. Disturbance measure model will be designed in the BTE domain. Suitable Genetic based model will be configured to catch the poles of the disturbance function. Poles of the model are points in time at which the model indicates peaks in disturbance. It is supposed that the poles happened at the time boundaries of phonemes.

3. Best Tree Encoding

BTE is a simple on/off entropy mapping of the signal into the bands in which the signal is decomposed using wavelet packets. The key property in BTE is the alignment of the neighboring frequency domain bands in wavelet packets decomposition of the signal. Adjacent bands are much closer in distance than the non adjacent bands.



Part a: Before BTE



Part b: After BTE

Figure 2: BTE bands are aligned such as to make adjacent wavelet bands are closer in distance than non adjacent bands.

Figure 2-a illustrates how bands are sorted according to Matlab wavelet packets function. Figure2-b indicates how bands are encoded in BTE. Bands are rearranged for calculating the BTE of the frame. The tree is Encoded into a single number that held information of tree structure {leaves} and weight according to figure 2-b.

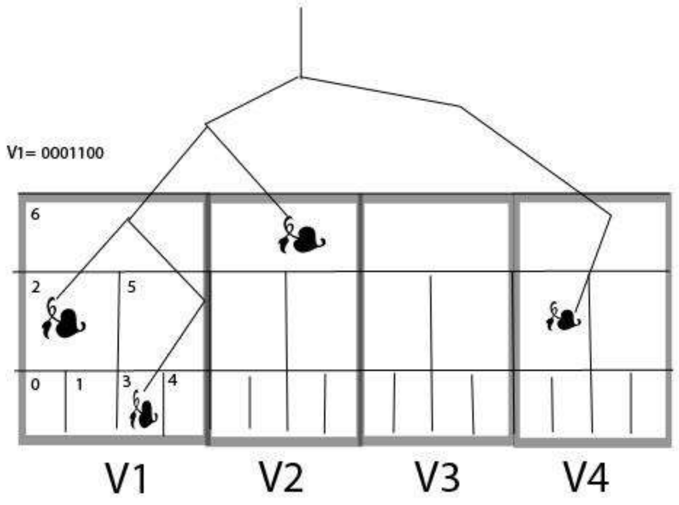


Figure 3: BTE for certain wavelet packets Best tree structure

The indicated tree structure in figure 3 will be encoded into features vector of 3 elements as shown in table 1.

TABLE 1

BEST TREE 4 POINT ENCODING EVALUATION.

Element	Binary Value	Decimal value	Frequency Band
V1	0001100	12	0 - 25 %
V2	1000000	64	25% - 50%
V3	0000000	0	50%-75%
V4	0000100	4	75%- 100%

Features BTE vector ζ for this example of speech frame will be $\zeta = \begin{bmatrix} 12 \\ 64 \\ 0 \\ 4 \end{bmatrix}$

4. DISTURBANCE MEASURE

BTE is implemented to map phoneme properties in the frequency domain. As discussed earlier, the properties should be homogeneous during the phoneme duration. This is the criterion that is intended to be measured in BTE domain. Figure 4-c illustrates the continuous changing in Best tree structure along speech utterance. BTE is an encoding for Best Tree structure. The variation in the Best Tree Structure is measurable through BTE variations along speech utterance. In this section, a suitable measure will be designed to model BTE variations.

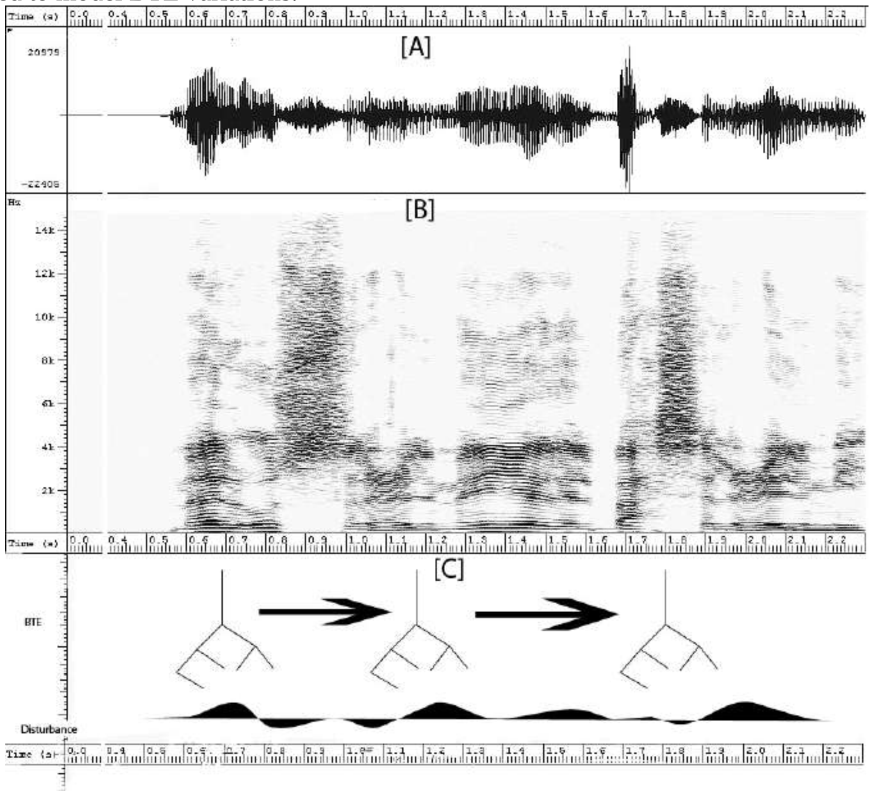


Figure 4: A) Time waveform of speech signal, b) Spectrogram of speech signal, c) Sketch of BTE encoding and associated disturbance.

Equation 1 is the template of the proposed disturbance measure function. It is required to find the suitable weights to get the optimal phoneme boundaries.

$$f(n) = \sqrt{\sum_{i=1}^{4} w_i \cdot \left(x_n^i - x_{n-1}^i\right)^2} \qquad (1)$$

where

$f(n)$	: Disturbance at frame number n.
w_i	: Band weight of band number i.
x_n^i	: BTE vector's component number i at frame number n.
x_{n-1}^i	: Depth of wavelet packets decomposition analysis.

As being illustrated earlier through the derivation of BTE vector, each vector's component in BTE is reflecting part of the Bandwidth of the signal. Speech signal is a stream of information encoded into signal frequency; this makes it logical that BTE vector's components should be weighted, as information is not equally distributed along the bandwidth.

Genetic algorithm is utilized to solve this optimization problem. The cost function is given by eqn.1. Given suitable number of labeled speech streams, it is possible to obtain the best set of weights for this optimization problem.

Below is the Matlab function " dist" of calculating the disturbance function for a given weight vector. This function is intended to be optimized using genetic process to find the minimum disturbance on a given training data. The output of the genetic process is the weight vector that gives the minimum disturbance.

```
%%[y] = dist(A,W)
%% A           : Columns based BTE vectors. It is 4 x L. 4 is the order of BTE vector. L is the %% length
in frames of the given Data.
%% W           : Weights array. It is 1 X 4 vector.
%% Y           : This is the output disturbance function. It is 1 X L vector. The first element %% is
calculated as the average value of the first 10% elements in the Y vector. This is to
%% avoid the discontinuities at the beginning of the array.
%% Amr M. Gody

function [y] = dist (A,W)
nbIn = nargin;
if nbIn < 1 ,    error('Not enough input arguments.');
    elseif nbIn == 1,     W = ones (size(A,1),1);
end;
W = Tocolumn(W);
L = size(A,2);
y = zeros ( 1,L);
for k=2:L
    try
    v2 = single( A(:,k));
    v1 = single(A(:,k-1));
    y(k) = sqrt(sum(W .* (v2 - v1).^2));
    catch
        m = int32( 0.1 * L);
        y(k) = mean(y(2:m+1));

    end
end
m = int32( 0.1 * L);
y(1) = mean(y(2:m+1));
end
```

5. PHONE LOCATIONS USING GENETIC ALGORITHM

In this section the genetic algorithm model for solving phone locations will be illustrated. Genetic algorithm is used to maximize of minimize certain cost function. The cost function should be parameterized in some parameters that are required to be minimizing or maximizing the cost function. Those parameters should be related to the intended target.
Recalling EQU. 1, disturbance is a function of the parameters array *w*. Recalling the following points

1- Phone segment is a homogenous part of speech utterance.
2- Phoneme Identification Information is not normally distributed over the bandwidth. Each phone has a different pattern of identification information that best identifying the phone.
3- BTE is a 4 elements vector. Each element reflects a 25% of the bandwidth of the signal.

Considering the above three points we can conclude the following
1- It is expected that disturbance is a minimum or ideally zero during the single segment.
2- Transition from phoneme to another phoneme may not necessary require equally changing value in all BTE vector's elements.

According to our intended target which is finding the locations of phoneme transitions, this may be obtained if we got the optimal ***w*** vector that minimizes the disturbance function. It is assumed that if the signal is a single phone, this should be zero disturbances. The following assumptions are considered

$$\sum_{i=1}^{4} w_i = 1 \qquad (2)$$

$$w_i \geq 0.01 \qquad \forall i; 1 \leq i \leq 4 \qquad (3)$$

Eqn. 2 and 3 are needed to exclude the forbidden solution of w = [0 0 0 0] from the set of possible solutions that may be obtained by genetic Algorithm. Eqn. 2 hold a meaning that the weight of BTE vectors are sum to one which means that it is considered that the information is 100% unit distributed over 4 parts.
Now it is needed to design a fitness function to evaluate the fitness of each solution during the optimization process of genetic algorithm. Each solution is a chromosome. In our case it is the ***w*** vector. The following function is supposed to be the fitness of the chromosome.

$$\Psi = \frac{\sum_{n} f(n)}{N} \qquad (4)$$

$f(n)$ is the disturbance function which is given in eqn.1. Ψ will give the mean of the disturbance. The best chromosome is that one gives the best mean of the disturbance function for a given BTE vectors.

One can ask , what is the benefit of establishing the w vector and why we do not make the disturbance without applying the weight vector w. The answer is to exclude the non informative parts of BTE vector from the calculations of the transitions. The non informative parts will impose a noise that is not needed for the process. The information is not normally distributed as indicated before. Also one cannot assume certain parts to contain the information. It is complex to be determined. So be using such heuristic technique like genetic, the problem may partially solved. The transitions between phones are estimated.

Here is below, the Matlab function "*fitv*" for calculating the fitness of the Genetic population.

```
function [val] = fitv(x)
    global buff; %This is a global buffer to store the array of bte vectors

    val = mean(dist (buff,x));

end
```

Below is the script of command line to call the genetic algorithm for optimizing the disturbance function.

```
% Evaluating the best weights for moving distance
    Ae = [ 1 1 1 1];
    Be = [1];
    w = ga(@fitv,4,Ae,Be,[],[],[0.01;0.01;0.01;0.01],[]);
```

Array Ae and Be are the constraint arrays. This will make "ga" to find solutions that satisfy the given constraints

$$A_e \times w = B_e \qquad (5)$$

Egn. 5 is used to pass the constraint given by eqn.2 to the built in Matlab function GA, which is used to run the genetic algorithm on the fitness function. The other constraint given by eqn.3 is directly passed to GA using the array [0.01; 0.01; 0.01; 0.01]. The empty square brackets are to pass default values arrays for the GA function. They are not used here in this research.

6. Results

The results of this research will be illustrated through examples. The following Matlab script is used to apply the procedure. Figure 5 gives a snapshot of the process. Y axis is used as a reference for defining the graph in the composite drawing. The legend is given by:

Draw with y axis value less than 0.5 → Speech waveform signal.
Draw with y axis value around 0.5 → Disturbance function.
Draw with y axis value = 1.0 → Obtained phoneme markers.
Draw with y axis value = 2.0 → Reference phoneme markers and annotations.

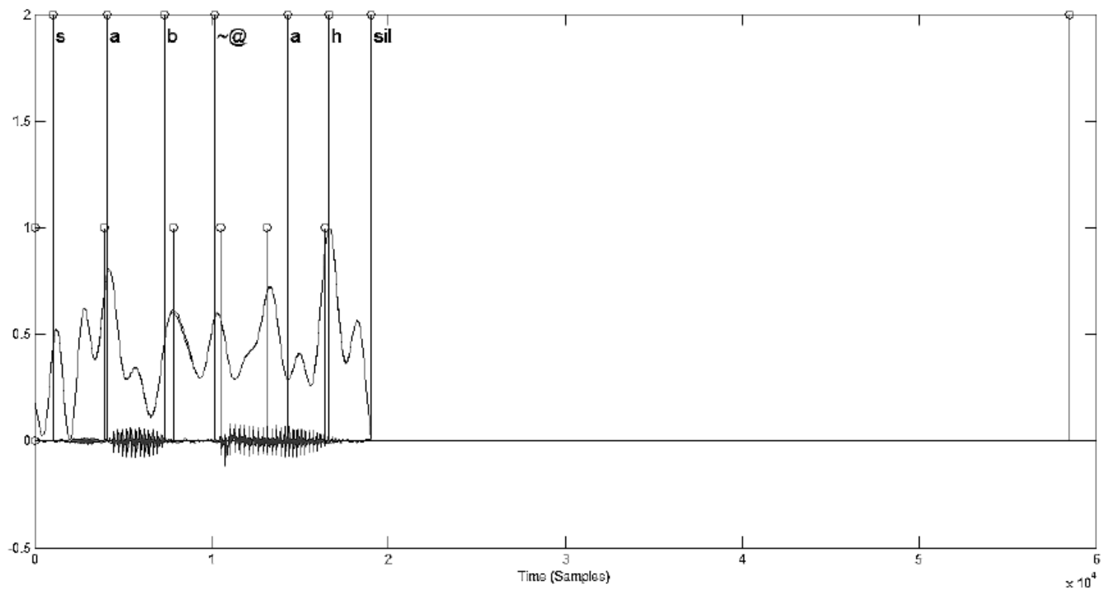


Figure 5 Composite draw of speech signal, disturbance function, Obtained phoneme boundaries, Reference phoneme boundaries and phoneme annotations respectively according to the y axis reference value.

As shown in figure; the obtained markers are very close to the reference markers. Local peaks are detected. Peaks detection process is applied on the disturbance curve to put phone markers. This task is very open to find the best peaks that best match with the reference markers. It is chosen here in this research that peaks should satisfy the local maximum condition of 20(ms) clearance area around the local peak. Clearance area is defined as area with no peaks. This can be modeled mathematically as the following equation

$$\upsilon = Local\_Max_{index}\{f(n)\}_{Grouped\_in\_40(ms)} \quad (6)$$

$f(n)$ is the disturbance function. If it is grouped in 40(ms) subgroups, it will make $\frac{N}{2}$ sub groups (N is the number of frames). In our case, frame length is 20(ms), this leads to groups of 2 elements will be considered to calculate the local peaks. This will discard the situation of two successive peaks are encountered. This is in other word indicates that the case of two major changes happened within 40(ms) will be discarded.

The error is calculated by applying the cross correlation process on the obtained markers (signal x) and the reference markers (signal y). In case of full coincidence of the locations of the two signals, this will make a maximum correlation at $\tau = 0$ in eqn.7. This makes τ gives an indication of how much is the mismatching maximum locations of the two signals.

$$\Re(\tau) = \sum_{\tau=-\infty}^{\infty} x(\tau) \cdot y(t-\tau) \tag{7}$$

Below is the Matlab function to calculate the error figure.

```
function err = GetErr(refv,v)
 x = xcorr(refv,v);
 u = find(x == max(x));
 refx = max(size(refv));
 err = abs(refx-u)/refx;
end
```

Applying the error figure estimation function on the available training data gives an average of ±8% drift off the reference locations. This error estimation does not include the effect of wrong markers. Wrong markers are defined as those markers exist in the result but don't exist in the reference or vice verses.

More examples are illustrated in Figure 6 to give more clarification of the peak peaking process and the locations of the obtained markers and the reference markers.

Below is the script of the Matlab function that implements the overall process of finding phoneme locations based on BTE. The function takes one parameter which is the file name of the speech signal. It expects to find HTK[1] format file of the annotations for the given file in the same folder. Below is a part of an HTK formatted file for labeling certain utterance.

```
308400 1550000 Sa
1550000 2693900 i
2693900 4298200 f
4298200 5909500 r
```

The first column indicates the beginning of the segment and the second column indicates the end of the segment. Numbers are in term of 100(ns). For example 308400 means segment (Sa) will start at $308400 \times 100 \times 10^{-9} = 0.0308(\text{sec})$.

The Matlab function "*findphonelocations*" reads the given speech signal in WAV[2] format. As mentioned earlier that the function expects to find an associated annotation file in the same folder with exact name as the given file but with file extension is ".LAB". The function uses a global buffer to pass the BTE vectors of the given file to the genetic algorithm fitness function. The global buffer is named "buff"

```
function [x] = findphonelocations(file)
    global buff;


    try

    [data fs] = wavread(file);
% Encoding frames into BTE3
    y = framing(data,20e-3*fs,0,0);
    a = bte3(y);
    buff = single (a);

% Read the associated Label file in HTK format
    labfile = strrep(file, 'wav', 'lab');

     [A l] = importhtkfile(labfile);
     reft = [A(:,1);A(size(A,1),2)] .* fs;
     refM = ones(size(A,1)+1,1) .* 2;

% Estmating the best weights for moving distance using genetic algorithm
    Ae = [ 1 1 1 1];
```

[1] HTK : Hidden Markov Model Toolkit http://htk.eng.cam.ac.uk/.

[2] WAV: Standard format for storing Audio signal. It is simple Pulse Coded Modulation samples. In this research speech signals are sampled at 32(KHz). Samples size = 16 (bits).

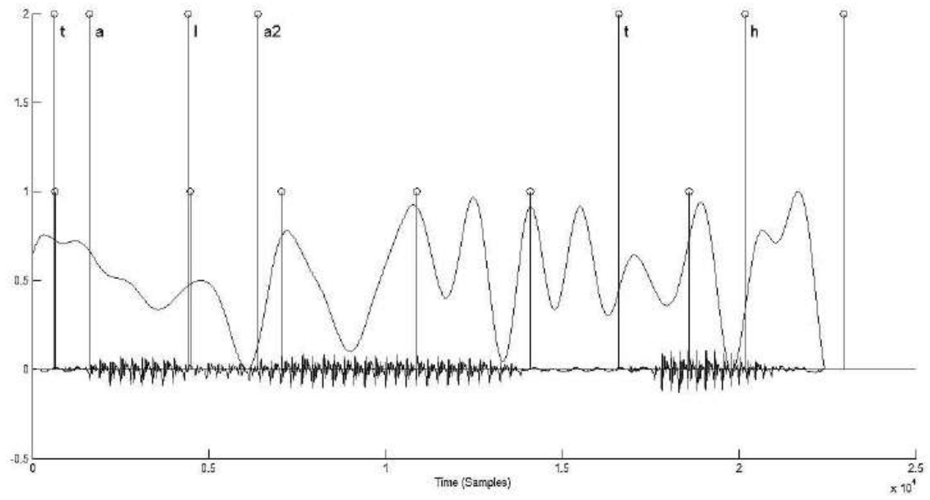
```
    Be = [1];
    w = ga(@fitv,4,Ae,Be,[],[],[0.01;0.01;0.01;0.01],[]);

% Evaluating the disturbance function for a given signal using the estimated weights.
    df = dist(buff,w);



%Find peaks in moving local durations of 20 (ms)

     [x p] = findpeaks (df,'minpeakdistance',2);%,'minpeakdistance',2);
     xpr = x ./ size(df,2);
     x = xpr .* size(y,1);

% Get Error Estimate
     scale = max(size(data))/max(size(reft));
     RefV = resize ( reft, scale);
     scale = max(size(data))/max(size(x));
     TestV = resize( x,scale);
     err = GetErr(RefV,TestV);

%Draw the results
     y1 = p > 1;
     hold on;
     y = normalize(data);
     plot(y);
     xlabel ('Time (Samples)');
     stem(x,y1);
     stem(reft,refM);
     dx = 0.01 * max(x);
     dy = 0.01 * max(refM);
     for k =1: max(size(l))
        text(reft(k)+ dx,refM(k)-5 * dy,l(k),'FontSize',14);
     end
     y2 = normalize (resample(df,max(size(data)),max(size(df))));
     t = 1:max(size(y2));
     plot(t,y2./max(y2),'r');
     hold off;
    catch exception
        display(exception.identifier);
    end
end
```

Here is below the Matlab function "resize", which is used to resize the vectors without changing the relative indices of the data. This function is very important to unify the size of the data vectors before applying the error function.

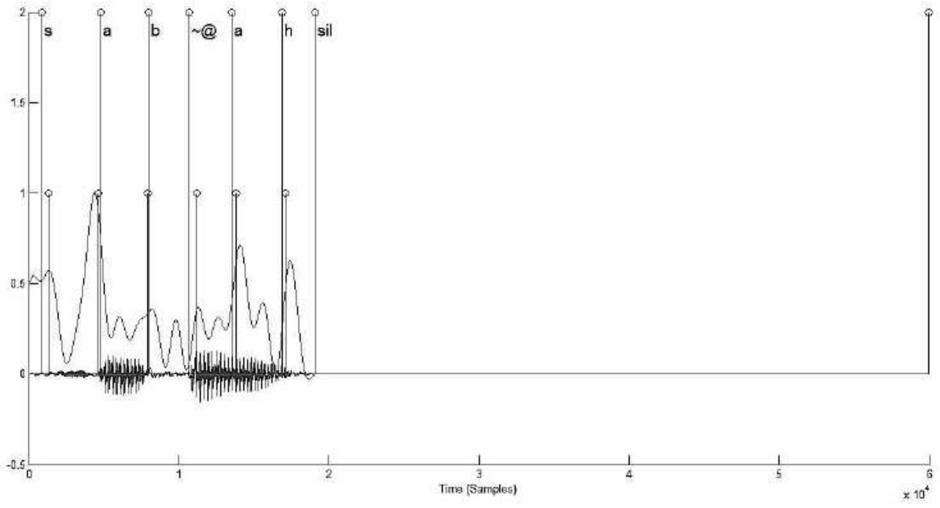
```
%%y = resize(x,scale)
% Resize array x using the given scale. All elements are mapped as of the
% percentage distance of thier index in the given array.
% x     : The array to be processed.
% scale : Index scale. If scale > 1, then the output array will be sized as
% (the original array size - 1) * scale
% y     : The output array.
% Amr M. Gody
function y = resize(x,scale)
oldsize = max(size(x));
newsize = (max(size(x))-1) * scale;
z = zeros(newsize,1);
for u = 1:oldsize
    d = int32( 1 +  (u-1) * scale);
    z(d) = x(u);
end
y = z;
end
```

Here is below the Matlab function for framing the speech signal

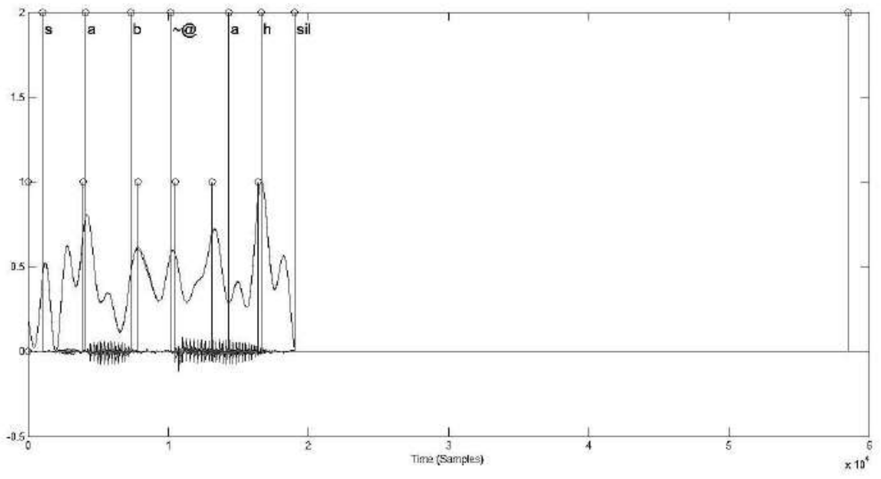
```
% data          :   Samples of Speech signal stored into a column array
% n             :   Required Frame size in samples
% p             :   Required Frames overlap in samples
% f             :   This is the output Frames array. The frames are stored into the columns.
% window        :       Default =0
%                      0   :   none
%                      1   :   Hamming
function [f] = framing(data,n,p,window)
    nbIn = nargin;
    if nbIn < 3 ,    error('Not enough input arguments.');
    elseif nbIn==3 , window = 0;
    end

    data = ToColumn(data);
i =size(data);

    maxindex = i(1);
    s=1;
    e=n;
    if window == 0 ,    f=data(s:e);
    elseif window ==1 , f=data(s:e).*hamming(n);
    end
    while i(1)>0;
        s=s+n-p;
        e=s+n-1;
        i(1)=i(1)-n+p;
        if(e<maxindex)

            if window == 0 ,    f = [f data(s:e)];
            elseif window ==1 , f = [f data(s:e).*hamming(n)];
            end
        end
    end
end
```



Sample a



Sample b



Sample c

Figure 6 BTE phoneme boundaries for two speech signals. Y axe is used to reference the signals in the composite graph. The signal at level 2 is the baseline markers signal obtained manually for phoneme boundaries. Annotation is available at the same level y=2. The obtained results are at Y = 1. It is highlighted by the small circles on the top of the marker lines. The disturbance measure function is fluctuating about Y = 0.5. Speech signal is fluctuating about Y = 0.0.

7. CONCLUSIONS

BTE is very promising in tracing non stationary property variations along speech duration. It is utilized here to find phoneme boundaries. It indicates promising results. Tracing a very hard to detect phoneme is obtained in reasonable accuracy. To avoid wrong markers, peak detection algorithm of disturbance function may be modified to catch the proper peaks. But for the detected peaks there are less than 10% drift of the manually estimated phone boundaries.

8. REFERENCES

[1] Amr M. Gody,"Wavelet Packets Best Tree 4-Points Encoded (BTE) Features", the 8th Conference on Language Engineering, PP. 189-198, 2008, Cairo, Egypt.

[2] Amr M. Gody," Voiced/Unvoiced and Silent Classification Using HMM Classifier based on Wavelet Packets BTE features ", the 8th Conference on Language Engineering, PP. 199-213, 2008, Cairo, Egypt.

[3] Iosif Mporas, Todor Ganchev, Nikos Fakotakis, "Phonetic segmentation using multiple speech features", International Journal of Speech Technology, Springer Netherlands, Volume 11, Number 2 / June 2008, PP. 73-85

[4] Kris Demuynck, Tom Laureys, "A Comparison of Different Approaches to Automatic Speech Segmentation", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 2448/2006, ISBN 978-3-540-44129-8, PP. 385-406

[5] Z. M. šarić, S. R. Turajlić, "A new approach to speech segmentation based on the maximum likelihood", Journal of Circuits, Systems, and Signal Processing, Birkhäuser Boston, Volume 14, Number 5 / September 1995, PP. 615-632

[6] Chin-Teng, Der-Jenq, Rui-Cheng, Gin-Der, "Noisy Speech Segmentation/Enhancement with Multiband Analysis and Neural Fuzzy Networks", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 2275/2002, ISBN 978-3-540-43150-3, PP. 81-94.

[7] Yanxiang Chen, Qiong Wang, "A Speaker Based Unsupervised Speech Segmentation Algorithm Used in Conversational Speech", Lecture Notes in Computer Science, Springer Berlin / Heidelberg, Volume 4798/2007, ISBN 978-3-540-76718-3, PP. 396-402.

Printed-**A**rabic **L**arge **TE**xt **C**orpus for OCR Research (P-ALTEC)

Waleed Fakhr[*1], Mohsen Moftah[*2], Mohsen Rashwan[**3], Mohamed ElMahallawy[*4]
[*] *College of Computing, Arab Academy for Science and Technology*
Ahmed Ismail street, Heliopolis, Cairo, Egypt
[1]waleedf@aast.edu
[2]mohsen.moftah@barmagyat.com
[4]mahallawy@aast.edu
[*] *Communications Department, College of Engineering, Cairo University*
Giza, Cairo, Egypt
[3]mohsen_rashwan@rdi-eg.com

***Abstract*—A Printed-Arabic Large Text Corpus (P-ALTEC) is proposed and developed by the Arabic Language Technology Center (ALTEC) [1]. The corpus is a large database that is intended for advanced research and prototype product development of optical character recognition (OCR) Arabic printed text. The database is designed to cover all important fonts used in Arabic printed documents (Windows and MAC), with plain and bold modes for each font, as well as a coverage of 7 sizes from 10 to 22. The database also has each document in clean, copied, mobile camera and digital camera-captured and typewriter forms to cover different practical qualities and acquisition states. The Arabic words come from a corpus that is a cleaned part of the Arabic Gigaword, with a lexicon of more than 500,000 unique words. An algorithm was developed to produce unique lists of words such that they cover all possible character and ligature shapes for all used fonts, at least 10 times in the clean data for each unique list. With 85 unique lists used, the database contains approximately 100,000 unique words, 11,000 pages, and 300 different character and ligature shapes. Character level segmentation is available for about 10% of the documents, with full word-based annotation and segmentation for the whole data. (P-ALTEC) should be available to the research community by March 2011.**

***Index Terms*— Arabic OCR, Arabic printed text, ALTEC, P-ALTEC**

I. INTRODUCTION

Arabic characters are used in writing by nearly a billion people worldwide. Hence, huge amount of printed Arabic documents are available without digital source. These documents may be governmental, historical, educational, and artistic such as books and magazines. Optical Character Recognition (OCR) is a technology that allows transforming such documents into digital forms for archiving, retrieval, and editing. OCR has matured quite well in European languages such as English and French [2-4], where the performance of available products is highly accurate and robust.
Arabic OCR products, on the other hand, still suffer from an obvious gap in performance with their Latin counterparts. In particular, in cases where the documents that are being processed are of low quality due to the aging of the source (books, magazines, historical documents), or the acquisition method (copier, mobile camera, Fax, etc.). Furthermore, Arabic characters have many different shapes that are font dependent both in single characters and in ligatures. Added

with the large number of fonts that are used in the documents (current and historical), this gives the Arabic OCR its special challenges and difficulties.

Since all OCR products depend on machine learning concepts, and since all machine learning algorithms require training data, then, a training database is always a very important component in producing any OCR product or research prototype. Designing a training database is always a difficult task, in particular, when the aim is to make products. In that sense, the OCR research team in ALTEC [1] spent a large amount of time and efforts trying to come up with an Arabic printed text database that is diverse enough to cover almost all different alternations of the Arabic characters and ligatures, over most frequent qualities encountered in practice. The database aimed covers approximately 300 different character and ligature shapes over 15 different Arabic fonts for Windows and MAC systems. The documents are available in clean form (print and scan), as well as copied form (copy from clean then scan), and digital and mobile camera forms. Also, a large number of books and theses covering a large time span are included in scanned forms using book digitizer technology. All the documents are transcribed down to the word level. As well, 10% of the documents (distributed over the whole document population) have character-ligature level segmentation and annotation. The database will be available to the research and industry by the end of March 2011, and will include approximately 11,000 pages, with more than 100,000 unique words and 15 different fonts, with 7 sizes between 10 and 22.

II. Arabic OCR Technology

Since the mid-1940s researchers have carried out extensive work and published many papers on character recognition. Most of the published work on OCR has been on Latin characters, with work on Japanese and Chinese characters emerging in the mid-1960s. Although almost a billion of people worldwide, in several different languages, use Arabic characters for writing (alongside Arabic, Persian and Urdu are the most noted examples), Arabic character recognition has not been researched as thoroughly as Latin, Japanese, or Chinese and it has almost only started in the 1970's. This may be attributed to the following:

(i) The lack of adequate support in terms of journals, books, conferences, and funding, and the lack of interaction between researchers in this field.

(ii) The lack of general supporting utilities like Arabic text databases, dictionaries, programming tools, and supporting staff.

(iii) The late start of Arabic text recognition.

(iv) The special challenges in the characteristics of the Arabic script as stated in the following section. These characteristics results in the fact that the techniques developed for other writings cannot be successfully applied to the Arabic writing: Different fonts, etc;

In order to be competent with the human capability at the digitization of printed text, font-written OCR's should achieve an omni-font performance at an average WER≤3% and an average speed ≥ 60 words/min. per processing thread. While font-written OCR systems working on Latin script can claim approaching such measures under favorable conditions, the best systems working on other scripts, especially cursive scripts like Arabic, are still well behind due to a multitude of complexities [windows magazine 2007]. For example, the best reported ones among the few Arabic omni font-written OCR systems can claim assimilation WER's 3%and 10% generalization WER's under favorable conditions (good laser printed windows and Mac fonts) [5-7].

Arabic OCR challenges

The written form of Arabic language while written from right to left presents many challenges to the OCR developer. The most challenging features of the Arabic orthography are [5-8]:

i) The connectivity challenge

Whether handwritten or font written, Arabic text can only be scripted cursively; i.e. graphemes are connected to one another within the same word with this connection interrupted at few certain characters or at the end of the word. This necessitates any Arabic OCR system to not only do the traditional grapheme recognition task but do another tougher grapheme segmentation one (see Figure 1) To make things even harder, both of these tasks are mutually dependent and must hence be done simultaneously.

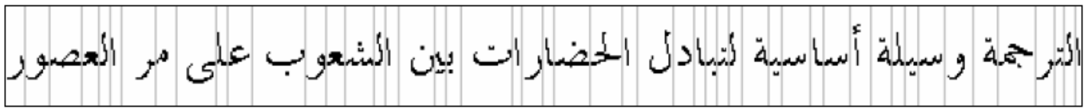

الترجمة وسيلة أساسية لتبادل الحضارات بين الشعوب على مر العصور

Figure (1): Grapheme segmentation process illustrated by manually inserting vertical lines at the appropriate grapheme connection points.

ii) The dotting challenge

Dotting is extensively used to differentiate characters sharing similar graphemes. According to Figure (2), where some example sets of dotting differentiated graphemes are shown, it is apparent that the differences between the members of the same set are small. Whether the dots are eliminated before the recognition process, or recognition features are extracted from the dotted script, dotting is a significant source of confusion – hence recognition errors – in Arabic font-written OCR systems especially when run on noisy documents; e.g. those produced by photocopiers.

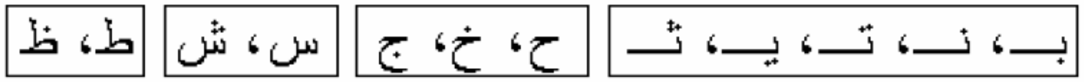


Figure (2): Example sets of dotting-differentiated graphemes

iii) The multiple grapheme cases challenge

Due to the mandatory connectivity in Arabic orthography; the same grapheme representing the same character can have multiple variants according to its relative position within the Arabic word segment {Starting, Middle, Ending, Separate} as exemplified by the 4 variants of the Arabic character " ع" shown in bold in Figure (3).

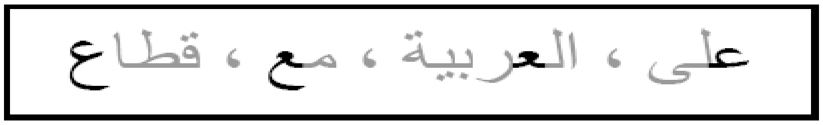


Figure (3): Grapheme “ ع” in its 4 positions; Starting, Middle, Ending & Separate

iv) The ligatures challenge

To make things even more complex, certain compounds of characters at certain positions of the Arabic word segments are represented by single atomic graphemes called ligatures. Ligatures are found in almost all the Arabic fonts, but their number depends on the involvement of the specific font in use. Traditional Arabic font for example contains around 220 graphemes, and another common less involved font (with fewer ligatures) like Simplified Arabic contains around 151 graphemes. Compare this to English where 40 or 50 graphemes are enough. A broader grapheme set means higher ambiguity for the same recognition methodology, and hence more confusion. Figure (4) illustrates some ligatures in the famous font “Traditional Arabic”.

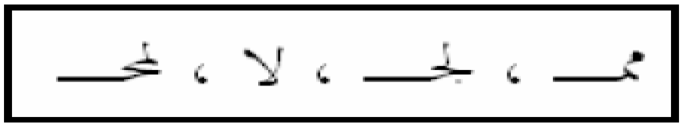


Figure (4): Some ligatures in the Traditional Arabic font.

iv) The overlapping challenge

Characters in a word may overlap vertically even without touching as shown in Figure (5).

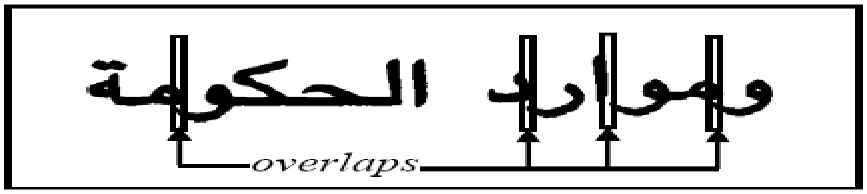


Figure (5): Some overlapped Characters in Demashq Arabic font.

v) Size variation challenge

Different Arabic graphemes do not have a fixed height or a fixed width. Moreover, neither the different nominal sizes of the same font scale linearly with their actual line heights, nor the different fonts with the same nominal size have a fixed line height.

vi) The diacritics challenge

Arabic diacritics are used in practice only when they help in resolving linguistic ambiguity of the text. The problem of diacritics with font written Arabic OCR is that their direction of flow is vertical while the main writing direction of the body Arabic text is horizontal from right to left (See Figure (6)). Like dots;

diacritics –when existent– are a source of confusion of font-written OCR systems especially when run on noisy documents, but due to their relatively larger size they are usually preprocessed.

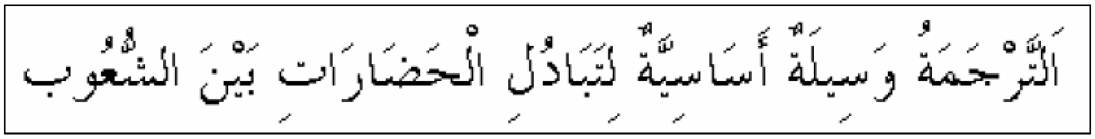

اَلتَّرْجَمَةُ وَسِيلَةٌ أَسَاسِيَّةٌ لِتَبَادُلِ الْحَضَارَاتِ بَيْنَ الشُّعُوبِ

Figure (6): Arabic text with diacritics.

III. Arabic OCR State of the Art

OCR is a highly mature technology for Latin script with excellent performance. The main challenges are in the pre-processing, page segmentation, speed of batch processing and post-processing. OmniPage-17 by **Nuance** is an example of such a product with less than 1% CER [9].

Regarding Arabic OCR products, there are mainly three mature engines from SAKHR, VERUS from NovoDynamics and ReadIRIS. Sakhr has around 1% CER for good quality documents but may drop significantly with poor quality documents. It offers high speed, and best output layout [10]. VERUS on the other hand is a little lower than Sakhr for good quality but significantly better for poor quality documents. It is to be noted that Bibliotheca Alexandrina uses both engines for its digitization project. Finally, ReadIRIS has the lowest performance of the three [10], however, no elaborate comparison between the three has been published yet.

Regarding the current research efforts, the focus mainly is on producing true Omni OCR for different font families, font sizes (specially the large), document pre-processing and framing, noise robustness, and batch-mode speed. Most recent research to improve the performance employ hidden Markov models (HMMs), and fusion between multiple OCR systems targeting Omni font performance. Recent competitions focus on document analysis and page segmentation (ICDAR 2009) [2].

From the above, for the Arabic OCR to narrow the gap in performance with the Latin counterparts, the following is required:

1- Creation of a standard thorough database that would be available to research and development groups who are doing serious Arabic OCR systems development.

2- Creation of a standard benchmark database that would be available to the research and development groups with regular evaluation workshops that include competitions on the benchmark data.

The aim of ALTEC is exactly the above. In that sense, the P-ALTEC database has been developed, where a portion will be dedicated to benchmarking and a near future competition.

IV. DATABASE DESIGN

1. Introduction

The project aims to generate a corpus or a database of wordlists and images to be used in OCR related development and research. The produced wordlists should cover every possible occurrence of Arabic letters and ligatures in different positions within the word. We used a cleaned portion of Arabic Gigaword corpus which contains 500K unique words as the source from which the required wordlists will be produced. The plan was to produce wordlists for 13 Arabic fonts namely Simplified Arabic, Arabic Transparent, and Traditional Arabic for Windows and Dahab, Riadh, and Naskh for MAC, each with Bold and normal, and also the normal Typewriter font. Each unique list is designed to cover every possible Arabic shape at least 10 times. Finally, we have extended the lists to 85 lists by having a unique list for each of the 7 font sizes used covering approximately 100,000 unique words.

2. Research Phase

During this phase many approaches were examined to tackle the problem. The first approach was to look at different letters' shapes within different context. Since we are looking at letter shapes, it is impossible to use the normal ASCII Code based character set, because this coding scheme keeps only one code for every letter called the code page, and the shape shown at display/print time is selected from the font page through a Contextual Analysis process, which determine which letter shape to display/print according to its position in the word. To handle letter shapes shape by shape, we had to use UNICODE Coding scheme that gives a unique code for every shape either a single letter or ligature as shown in Table1. Therefore, the first conclusion was that we have to convert the source wordlist from ASCII coding to UNICODE coding system. To guarantee that the wordlist covers all letters in all contexts, we decided to work on tri-graphemes where a tri-grapheme is a group of three letters. A prototype application was development that creates all possible tri-graphemes, then the program scans the wordlist and counts all occurrences of every tri-graphemes (determine tri-graphemes coverage); Tri-graphemes with counts greater than zero are considered legal. Running the prototype on a subset of the source wordlist, we found that working on Tri-graphemes will require a huge wordlist to select from to satisfy an acceptable coverage. In addition to that, the produced wordlist will be impractically large, so it was decided to work on bi-graphemes which will cover all possible latter context and produce a wordlist of practical size. The prototype was run on the source wordlist based on bi-graphemes to produce 3225 legal bi-graphemes out of 15625 possible bi-graphemes. Another program was developed to create output wordlist for the extracted legal bi-graphemes, the produced wordlist was 25K words for 20 words coverage which is still considered a large number.

Table 1: UNICODE Coding scheme

Hex Code	Shape	Hex Code	Shape	Hex Code	Shape	Hex Code	Shape	Hex Code	Shape
fe80	ﺀ	fe99	ﺙ	feb2	ﺲ	fecb	ﻋ	fee4	ﻤ
fe81	ﺁ	fe9a	ﺚ	feb3	ﺳ	fecc	ﻌ	fee5	ﻥ

fe82	ﺂ	fe9b	ﺛ	feb4	ﺴ	fecd	ﻍ	fee6	ﻦ
fe83	ﺃ	fe9c	ﺜ	feb5	ﺵ	fece	ﻎ	fee7	ﻧ
fe84	ﺄ	fe9d	ﺝ	feb6	ﺶ	fecf	ﻏ	fee8	ﻨ
fe85	ﺅ	fe9e	ﺞ	feb7	ﺷ	fed0	ﻐ	fee9	ﻩ
fe86	ﺆ	fe9f	ﺟ	feb8	ﺸ	fed1	ﻑ	Feea	ﻪ
fe87	ﺇ	fea0	ﺠ	feb9	ﺹ	fed2	ﻒ	Feeb	ﻫ
fe88	ﺈ	fea1	ﺡ	feba	ﺺ	fed3	ﻓ	Feec	ﻬ
fe89	ﺉ	fea2	ﺢ	febb	ﺻ	fed4	ﻔ	Feed	ﻭ
fe8a	ﺊ	fea3	ﺣ	febc	ﺼ	fed5	ﻕ	Feee	ﻮ
fe8b	ﺋ	fea4	ﺤ	febd	ﺽ	fed6	ﻖ	Feef	ﻯ
fe8c	ﺌ	fea5	ﺥ	febe	ﺾ	fed7	ﻗ	fef0	ﻰ
fe8d	ﺍ	fea6	ﺦ	febf	ﺿ	fed8	ﻘ	fef1	ﻱ
fe8e	ﺎ	fea7	ﺧ	fec0	ﻀ	fed9	ﻙ	fef2	ﻲ
fe8f	ﺏ	fea8	ﺨ	fec1	ﻁ	feda	ﻚ	fef3	ﻳ
fe90	ﺐ	fea9	ﺩ	fec2	ﻂ	fedb	ﻛ	fef4	ﻴ
fe91	ﺑ	feaa	ﺪ	fec3	ﻃ	fedc	ﻜ	fef5	ﻵ
fe92	ﺒ	feab	ﺫ	fec4	ﻄ	fedd	ﻝ	fef6	ﻶ
fe93	ﺓ	feac	ﺬ	fec5	ﻅ	fede	ﻞ	fef7	ﻷ
fe94	ﺔ	fead	ﺭ	fec6	ﻆ	fedf	ﻟ	fef8	ﻸ
fe95	ﺕ	feae	ﺮ	fec7	ﻇ	fee0	ﻠ	fef9	ﻹ
fe96	ﺖ	feaf	ﺯ	fec8	ﻈ	fee1	ﻡ	Fefa	ﻺ
fe97	ﺗ	feb0	ﺰ	fec9	ﻉ	fee2	ﻢ	Fefb	ﻻ
fe98	ﺘ	feb1	ﺱ	feca	ﻊ	fee3	ﻣ	Fefc	ﻼ

After discussion with font and OCR experts, we confirm that letters shapes do not change with context so we decided to work on uni-graphemes, and that it is enough to have 10 instances of each uni-grapheme i.e. coverage will be 10 words. Accordingly, the same prototype was run for uni-grapheme producing a significantly small wordlist of about 600 words. Since wordlists will be produced for different fonts, it was time to introduce ligatures for different fonts. Another program was developed to count the coverage of some commonly used ligatures, it was found that the coverage is dramatically bad; therefore, we had to consider the ligatures as part of the Arabic letter list and re-create the wordlist that satisfies the coverage required. The produced wordlist was about 1200 words.

The last part of the research phase was to examine producing different wordlists for different fonts. After carrying out statistics on the source wordlist and uni-grapheme and ligatures coverage, it was found that uni-graphemes and ligatures are not equally covered in the source word list so words for uni-graphemes/ligatures with lower coverage should be repeated among different lists to guarantee that all uni-graphemes and ligatures are represented in every list. Accordingly, the following algorithm, illustrated in Figure 7, was used to produce wordlists.

1. In the source wordlist, every word was assigned a rank initially zero.
2. Uni-graphemes/ligature are processed individually. And for every uni-grapheme/ligature based on the coverage required, words from the source wordlist are selected and sorted by rank on ascending order. The selected words are then copied to the output list and their ranks in the source wordlist is incremented by one.
3. Step 2 is repeated until all uni-graphemes/ligatures are processed.

Since at every iteration the selected words are sorted by rank in ascending order, it is guaranteed that always less used words and/or not used words come at the top of the list Figure 7 explains this algorithm.

In this way if the uni-grapheme/ligature has high coverage it will be represented by different words in different lists. Repetition of words in different lists (overlap) will increase with uni-graphemes/ligatures with lower coverage.

The previous steps were repeated as many times as the required number of lists.

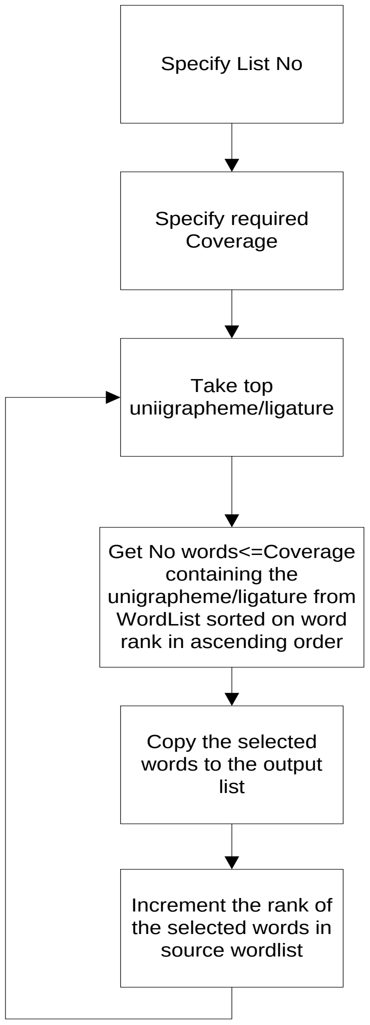

Specify List No
Specify required Coverage
Take top uniigrapheme/ligature
Get No words<=Coverage containing the unigrapheme/ligature from WordList sorted on word rank in ascending order
Copy the selected words to the output list
Increment the rank of the selected words in source wordlist

Figure (7): Word Selection Algorithm

Finally, the implementation of the above research resulted in developing a number of programs to carry out the required tasks to achieve the following:

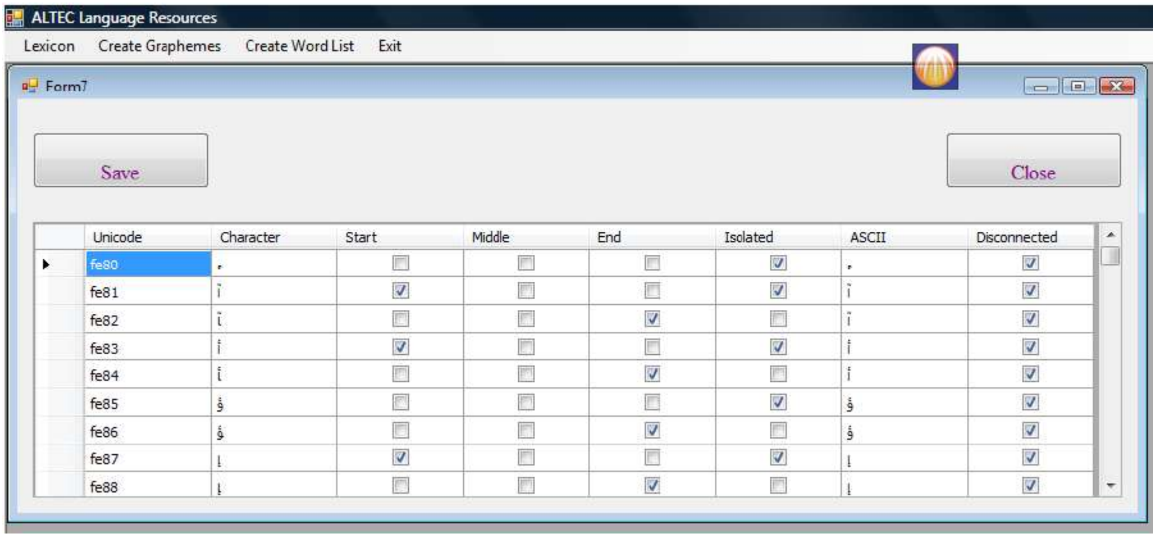
1. Use three Arabic fonts simplified Arabic, Arabic transparent, and Traditional Arabic. In two modes Normal, and Bold.
2. A different list will be used for different fonts and different mode.
3. Two sets of lists will be produced one for Windows and another for MAC.
4. Another list will be for typewriter font.

3. Development and Implementation Phase

The following are the steps followed to create the Arabic Wordlist needed. For every step there is corresponding program developed to carry out the specific stem

(i) Specify Arabic Letters Specifications

In this program all Unicode Arabic letters shapes are displayed and letter specification is set. The specification include the position of each letter and whether it is connected or not, Figure 8.



Unicode	Character	Start	Middle	End	Isolated	ASCII	Disconnected
fe80	ﺀ	☐	☐	☐	✓	ء	✓
fe81	ﺁ	✓	☐	☐	✓	آ	✓
fe82	ﺂ	☐	☐	✓	☐	آ	✓
fe83	ﺃ	✓	☐	☐	✓	أ	✓
fe84	ﺄ	☐	☐	✓	☐	أ	✓
fe85	ﺅ	☐	☐	☐	✓	ؤ	✓
fe86	ﺆ	☐	☐	✓	☐	ؤ	✓
fe87	ﺇ	✓	☐	☐	✓	إ	✓
fe88	ﺈ	☐	☐	✓	☐	إ	✓

Figure 8: Unicode Arabic Letters Specifications

(ii) Create Unicode Lexicon

This program reads the words from the pre-prepared Arabic wordlist and each word undergoes a contextual analysis process to convert the word from normal ASCII code to Unicode shapes. The program analyses the word letter by letter to determine the specification of each one based on its position in the word and whether it is connected or not.

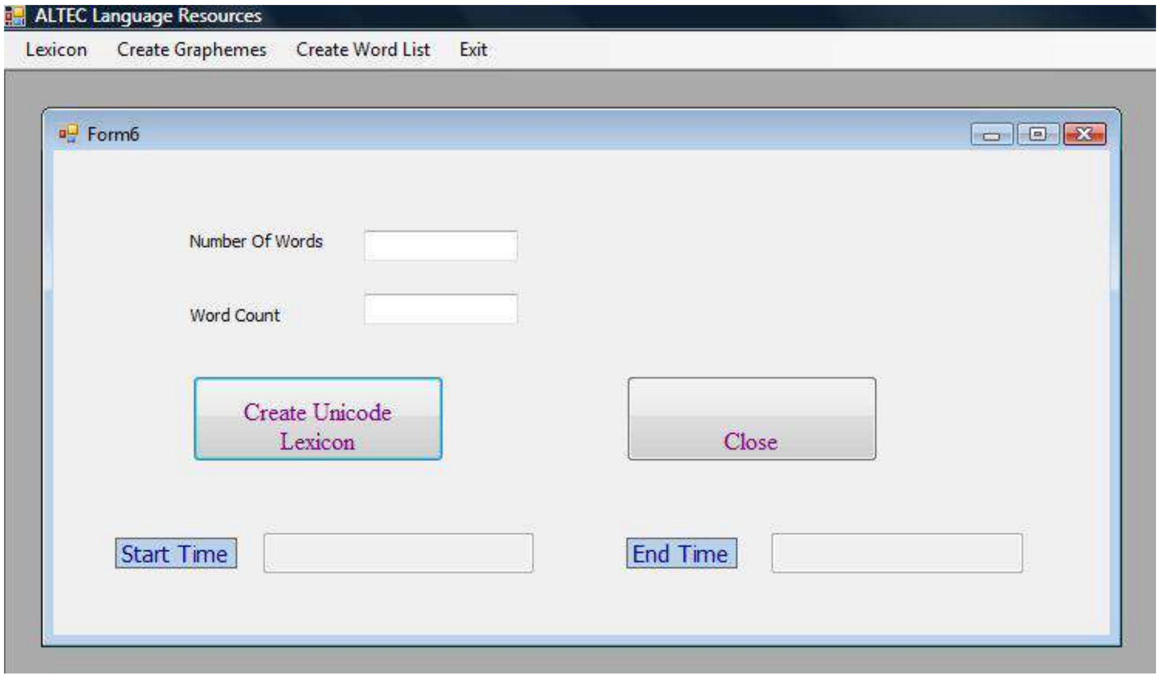


Figure 9: Create UNICODE Lexicon

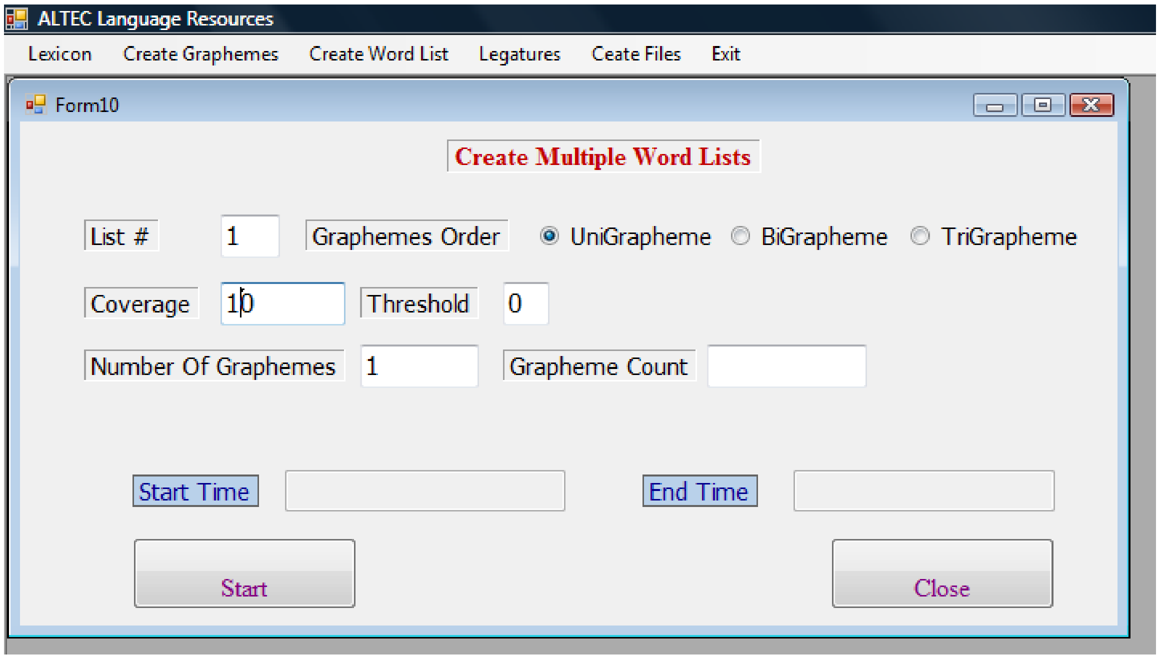
The result is table containing the words with its Unicode representation. Table 2 shows a sample of the UNICODE lexicon produced. As shown, using UNICODE scheme enables us to deal with letters shapes separately.

Table 1: Unicode and ASCII Wordlist

Unicode		ASCII	
ﻟ ﺯ ﺭ ﺍ ﻋ ﺔ	لزراعة	ل ز ر ا ع ة	لزراعة
ﻟ ﺯ ﺭ ﺍ ﻋ ﺗ ﻬ ﺎ	لزراعتها	ل ز ر ا ع ت ه ا	لزراعتها
ﻟ ﺯ ﻋ ﻳ ﻣ ﻧ ﺎ	لزعيمنا	ل ز ع ي م ن ا	لزعيمنا

(iii) Create Multiple Wordlist

This program produces as many wordlists as required based on the input criteria as shown in Figure 10. The produced wordlist order could be based on Tri-graphemes, bi-graphemes, or uni-graphemes, below is the explanation of the input criteria.


ALTEC Language Resources

Lexicon | Create Graphemes | Create Word List | Legatures | Ceate Files | Exit

Form10

Create Multiple Word Lists

List # 1 | Graphemes Order [✓ UniGrapheme] BiGrapheme TriGrapheme

Coverage 10 | Threshold 0

Number Of Graphemes 1 | Grapheme Count

Start Time | End Time

Start | Close

Figure 10: Create Multiple Word List

Grapheme Order: Uni-grapheme, Bi-graphemes, or Tri-graphemes

Coverage: minimum number of instances of required grapheme.

Threshold: the grapheme frequency after which grapheme will be processed.

Number of Graphemes: the number of graphemes to be processed in the current run.

The selected words that satisfy the input criteria are written to a new list according to the algorithm in Figure 7.

The selected criteria in our run was:

Grapheme Order: Uni-grapheme

Coverage: 10

Threshold: 0

This program was run 13 times to produce 13 output wordlists (then was extended to produce 85 lists, but throughout this section we will mention the basic 13 lists). Every wordlist contains a number of unique words not included in any other wordlist, some words common with another wordlist, some words common with three wordlists and so forth. Table 3 and Figure 11 show a statistics of the word distribution among the 13 wordlists.

Table 2: Produced Lists and Repetition Distribution

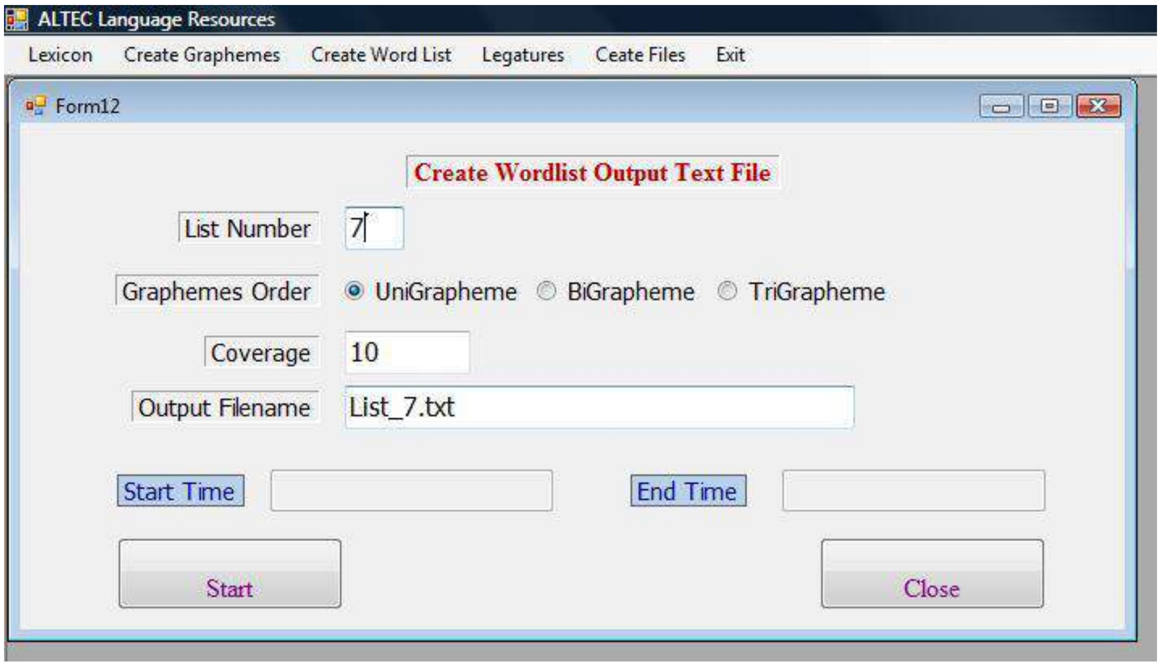
		Lists													
		List01		List02		List03		List04		List05		List06		List07	
		Count	%	Count	%	Count	%	Count	%	Count	%	Count	**%**	Count	%
Repetition	1	897	74.6	871	72.3	865	71.7	868	72.5	855	71.3	848	70.7	850	70.5
	2	49	4.1	66	5.5	90	7.5	105	8.8	118	9.8	115	9.6	124	10.3
	3	92	7.7	98	8.1	82	6.8	50	4.2	47	3.9	64	5.3	76	6.3
	4	44	3.7	45	3.7	52	4.3	55	4.6	59	4.9	51	4.3	36	3.0
	5	21	1.7	28	2.3	18	1.5	20	1.7	26	2.2	22	1.8	22	1.8
	6	10	0.8	13	1.1	16	1.3	10	0.8	10	0.8	12	1.0	10	0.8
	7	20	1.7	14	1.2	15	1.2	21	1.8	16	1.3	18	1.5	19	1.6
	8	5	0.4	5	0.4	10	0.8	10	0.8	7	0.6	3	0.3	5	0.4
	9	15	1.2	15	1.2	10	0.8	10	0.8	13	1.1	17	1.4	15	1.2
	10	10	0.8	10	0.8	10	0.8	10	0.8	10	0.8	10	0.8	10	0.8
	11	1	0.1	1	0.1	2	0.2	2	0.2	2	0.2	2	0.2	2	0.2
	12	9	0.7	9	0.7	8	0.7	8	0.7	8	0.7	8	0.7	8	0.7
	13	29	2.4	29	2.4	29	2.4	29	2.4	29	2.4	29	2.4	29	2.4
		1202		1204		1207		1198		1200		1199		1206	

Table 3 continued

		Lists											
		List08		List09		List10		List11		List12		List13	
		Count	%	Count	%	Count	%	Count	%	Count	%	Count	%
Repetition	1	841	68.6	848	70.7	849	69.5	847	69.1	800	66.7	821	67.2
	2	135	11.0	114	9.5	115	9.4	116	9.5	128	10.7	121	9.9
	3	80	6.5	82	6.8	87	7.1	105	8.6	86	7.2	86	7.0
	4	36	2.9	51	4.3	57	4.7	36	2.9	62	5.2	64	5.2
	5	25	2.0	28	2.3	19	1.6	18	1.5	28	2.3	25	2.0
	6	15	1.2	7	0.6	13	1.1	14	1.1	11	0.9	15	1.2
	7	15	1.2	18	1.5	18	1.5	21	1.7	15	1.3	21	1.7
	8	7	0.6	8	0.7	1	0.1	10	0.8	9	0.8	0	0.0
	9	13	1.1	12	1.0	19	1.6	10	0.8	11	0.9	20	1.6
	10	10	0.8	10	0.8	10	0.8	10	0.8	10	0.8	10	0.8
	11	2	0.2	2	0.2	2	0.2	2	0.2	1	0.1	1	0.1
	12	8	0.7	8	0.7	8	0.7	8	0.7	9	0.8	9	0.7
	13	29	2.4	29	2.4	29	2.4	29	2.4	29	2.4	29	2.4
		1216		1217		1227		1226		1199		1222	

(iv) Create Output Wordlist Files

This program takes as input the list number for which it is required to create wordlist file in text format. File name is automatically created, the rest of the input criteria specify the wordlist to be used as input (see Figure 11).



Create Wordlist Output Text File

List Number	7
Graphemes Order	[✓ UniGrapheme] BiGrapheme TriGrapheme
Coverage	10
Output Filename	List_7.txt
Start Time	
End Time	

Figure 11: Create Output Wordlist Files

This program is run 13 times to produce the required 13 wordlists.

(v) Create Text for Printing

The produced text file was opened using MS Word one at a time and the font and size were changed to produce different file for different font and size.

4. Request for Proposal Phase

The production of the database is currently underway based on a request for proposal (RFP) that has been distributed to all local companies working in the field. The details of the RFP are as follows.

(i) Fonts and Sizes for the word lists

The production of the required output will be carried out according to the following specifications:

1. Fonts:
    a. For Windows Platform
        1) Simplified Arabic

   2) **Arabic Transparent**
   3) **Traditional Arabic**

b. For MAC Platform:

   1) **Dahab**
   2) **Riadh**
   3) **Naskh**

Each font is done twice for both **Normal** and **Bold.**

c. Manual **Typewriter** (fixed mode and font)

(This sums to 13 different streams).

2. Sizes (for Windows and MAC): Each of the above is required to be produced for those sizes (except the typewriter): **10, 12, 14, 16, 18, 20, and 22**

The bidders will be provided with the following files:

1. Arabic Word Lists: these lists are text files named List01.txt, List02.txt, up to List85.txt.
2. Letters and Ligature Files: these files contain the shapes of individual Arabic letters and Arabic Ligatures for every font, the files names are:
   a. For Simplified Arabic: Simplified-Arabic_Letters-and-Ligatures.txt
   b. For Arabic Transparent: Arabic-Transparent_Letters-and-Ligatures.txt
   c. For Traditional Arabic: Traditional-Arabic_Letters-and-Ligatures.txt
   d. For Dahab: Dahab_Letters-and-Ligatures.txt
   e. For Riadh: Riadh_Letters-and-Ligatures.txt
   f. For Naskh: Naskh_Letters-and-Ligatures.txt

(ii) Books and Theses Documents:

1500 pages from different books will be selected (average of 10 pages from each book for copyright constraints which gives approximately 150 books). The books will be chosen to cover uniformly the past 50 years. In addition, 1000 theses (in Arabic) will be selected as well which should also cover uniformly the past 50 years. Books should come from at least 15 different categories based on the fonts and sizes used. The books used will be classified manually and approved by ALTEC. Theses should come from at least 10 different categories based on the fonts and sizes used. The theses used will be classified manually and approved by ALTEC.

(iii) Naming Convention

The output files names should follow the following naming convention:

The Base file names will consist of the following parts

- System Type: WIN (for windows), MAC, BK (for book), or TH (for thesis)
- Source List No: 01, 02, etc., or book No: 01, 02, etc.
- Page number in the list or in the book 01, 02, etc.
- Font Name: (SA for Simplified Arabic), (AT for Arabic Transparent), (TA for Traditional Arabic) (DH for Dahab, RH for Riadh and NH for Naskh).
- N for Normal, B for Bold.
- TP for Typewriter
- Font Size: 10-22

Example:

For Windows system, Source List file List02.txt, 3rd page in the list, font Simplified Arabic, Size 10, Normal the file name will be:

WIN_02_03_SA_10_N

For MAC system, Source List file List08.txt, 10th page, font Dahab, Size 10, Bold the file name will be:

MAC_08_ 10_DH_10_B

For the book number 17, 7th page, noting that the books and theses will be captured directly by a book digitizer as an image:

BK_17_07

(iv) Document Production

The documents production stage has two steps for the windows. MAC and typewriter lists: the first is the printing step and the second is the scanning step. As for the books and theses, we go directly to the scanning step.

<u>Printing and Imaging</u>

In printing step, the produced output files are printed then undergo different processes to add noise to the produced document. At the end of this step, the following document versions should have been produced:

1. Clean Version: the clean version is the first print out from the created files. Printing should be done using a different printer for every document set (at least 20 different printers should be used). In addition, the original document produced by typewriter is to be considered as clean version.
2. Copy Version: the clean version should be photocopied using different photocopying machines (at least 12).

3. Photo the clean version using Digital Cameras and Mobile Cameras. (At least 10 digital cameras and 10 mobile cameras). In this case, no scanning will be required since we will get the jpg images directly. All cameras should be at least 5Mpixel of resolution, and the distance to the documents should be 50cm. 50% of the imaging should be with separate cameras, and 50% with mobile cameras. The produced jpg images of this step will not undergo any further processing.

Scanning and Digitizing

The documents produced by the printing step (1 and 2 above) are scanned using a different scanner for every set of documents (at least 12 scanners are required), and saved in (jpg) format. The scanning should be done using the following resolutions: 200, 300 and 600 dpi.

As for the books and theses, a Book Digitizer must be used to produce three resolution versions of each page: 200, 300, and 600 dpi.

At this stage, all documents have been transformed into jpg images.

The produced (jpg) files names should have the same name as the source file from which it was printed in addition a new suffix to show the version from which it was created. The suffixes will be as follows:

C: Clean Version

P: Copy Version

D: Digital Camera Version

M: Mobile Camera Version

B: Books Version

H: Thesis Version

Another suffix should be added which indicates the resolution used.

Accordingly, if we print a file such as (WIN_05_03_TA_16_N) and expose it to different processing mentioned before, the scanned file names should be as follows:

Clean Version, 200 dpi: WIN_05_03_TA_16_N_C_2.JPG

Clean Version, 300 dpi: WIN_05_03_TA_16_N_C_3.JPG

Clean Version, 600 dpi: WIN_05_03_TA_16_N_C_6.JPG

Copy Version, 200 dpi: WIN_05_03_TA_16_N_P_2.JPG

Copy Version, 300 dpi: WIN_05_03_TA_16_N_P_3.JPG

Copy Version, 600 dpi: WIN_05_03_TA_16_N_P_6.JPG

Digital Camera Version: WIN_05_03_TA_16_N_D.JPG

Or a Mobile Camera Version: WIN_05_03_TA_16_N_M.JPG

As for the Typewriter written documents we will have the original printed version (clean) then they will undergo the copying step as well as the camera step. The naming will be as follows (also, for 200, 300 and 600 dpi):

Clean Version, 200dpi:	TP_13_05_C_2.JPG
Copy Version, 200 dpi:	TP_13_05_P_2.JPG

As for the Books and Theses names, the dpi should be added to the base name (2, 3 and 6 for 200, 300 and 600dpi):
BK_17_07_3.jpg (for book number 17, page number 7 and 300dpi).
Same goes for these:
TH_44_01_6.jpg (for thesis number 44, page number 1 and 600dpi)

(v) Segmentation and Annotation:

The following are the required outputs associated with each image file:

1- It is required to have a text transcription file for each image file. Text transcription files are required to be xml files that include all the details of each image (file name, font, size, quality, etc.) plus the full text transcription of the corresponding image for each line.
2- It is required to have full line segmentation for each line in the image with full information about the starting/end and height of each line in the image so that the user can extract the line directly from the image (using the upper right and lower left <x , y> coordinates).
3- It is required to have word boundary segmentation for all the lines, which gives the starting /end of each word, its height (using the upper right and lower left <x , y> coordinates).
4- This information should be included in the transcription files and it should allow the user to extract any word directly from the image.
5- It is required to have character boundary segmentation for a portion of the produced data. These boundary segmentations should also give the <x , y> coordinates of the box containing the character.
   The particular portion to be character-segmented documents should be 10% of the whole data, such that approximately 10% of the images are taken for each category as follows:
   a. 10% for each font (12 fonts), each size (7 sizes) and each resolution (3 resolutions) plus 10% of the typewriter images.
   b. 10% of all the copied documents images
   c. 10% of all the digital camera and 10% of all the mobile camera images.
   d. The pages taken should be taken at random to cover all shapes and characters.
   e. 10% of all the books images such that 10% is taken from each book category, and for each resolution.
   f. 10% of all the theses images such that 10% is taken from each category and for each resolution.

Transcription files naming should have the same naming convention as the corresponding image files, except for the xml extension for example:

WIN_05_03_TA_16_N_C_2.xml

1- The convention to be used in all segmentations is the box convention, where each line, word, character (ligature) would be enclosed by a box. The segmentation gives the (x1,y1) and (x2,y2) coordinates of that box, which are the upper right and lower left coordinates respectively.
2- Each line in the transcription files should correspond exactly to the same line in the corresponding image.

V. Conclusion

A Printed Arabic Text Large Database (P-ALTEC) is proposed and developed by the Arabic Language Technology Center (ALTEC). The database is intended for advanced research and prototype product development of optical character recognition (OCR) Arabic printed text. The database is designed to cover all important fonts used in Arabic printed documents (Windows and MAC), with plain and bold data for each font, as well as a coverage of sizes from 10 to 22. The database also has each document in clean, copied, camera-captured and typewriter forms to cover different practical qualities and acquisition states. The database contains approximately 100,000 unique words, 11,000 pages, and 300 different character and ligature shapes. Character level segmentation is available for about 10% of the documents, with full word-based annotation and segmentation for the whole data. (P-ALTEC) should be available to the research community by March 2011.

References

[1]. ALTEC website: http://www.altec-center.org/index.php
[2]. ICDAR 2009 technical program: http://www.cvc.uab.es/icdar2009/techprog.html
[3]. J. Makhoul et. al., "Multilingual Machine Printed OCR", IJPRAI (15.1) 2001.
[4]. Mohamed Cheriet et. al., "Character Recognition systems, A Guide for Students and Practitioners", Wiley Inter-science, 2007.
[5]. Rashwan, M., Fakhr, W., Attia, M., El-Mahallawy, M., Arabic OCR System
[6]. Analogous to HMM-Based ASR Systems; Implementation and Evaluation, Journal of Engineering and Applied Science, Cairo University, www.Journal.eng.CU.edu.eg, December 2007.
[7]. Al-Badr, B., Mahmoud, S.A., "Survey and Bibliography of Arabic Optical Text Recognition", Elsevier Science, Signal Processing 41 (1995) pp. 49-77.1.
[8]. Govindan, V.K., & Shivaprasad, A.P., "Character recognition A review", Pattern Recognition, Vol. 23, No. 7, 1990, pp. 671-683.
[9]. Hazem Y. Abdelazim, "Recent trends in Arabic OCR," in Proc. 5th Conference of Engineering Language, Ain Shams University, 2005.
[10]. http://www.nuance.com/imaging/omnipage/omnipage-professional.asp
[11]. H. Osman, Orange Egypt, personal communication.

ARABIC LINGUISTICS AND IN-DEPTH TEXT PROCESSING

Nabil Ali
nabilalii@gmail.com

Extended abstract **-Text processing is currently being developed into a level of computation much deeper than the conventional approaches adopted so far.**

Main motivations are:

- **The emergence of the Semantic Web**
- **Growing number of text-intensive applications, cultural and business as well.**
- **The increasing importance of language acquisition methods using ICT.**

These have led to a paradigmatic shift in text processing characterized by the following major trends:

- **from generative grammar to cognitive grammar**
- **from list-based lexicography to concept-based lexicology**
- **from taxonomical classification to computational ontology**
- **from parsing to understanding**
- **from ad-hoc textual output generation to systematic text generation and summarization**
- **from statistical vs-rule-based dichotomy to genuine hybridity**
- **from sentential processing to textual corpus computational techniques.**

All these trends call for more serious Artificial intelligence techniques capable enough to deal with linguistic complexities involved in textual computation.

The above-mentioned trends are overviewed from an Arabic language perspective.

The study attempts to map between brain cognitive functions related to indeterminacy to language tools and mechanisms to support such functions.

The study concludes by presenting list of proposed research areas in both fields of Arabic computational and theoretical linguistics.

The presentation will be delivered in Arabic.

Arabic Character Recognition Using statistical Moment Invariants and ANN

Ismail I. Amr*, Mohamed Amin**, Passent El-Kafrawy**1, and Amr M. Sauber**2

* College of Computers and Informatics

Misr International University, Cairo, Egypt

**Mathematics and CS Department, Faculty of Science,

Menoufiya University, Egypt

1passentmk@gmail.com, 2amrmausad@gmail.com

***Abstract* - Many Arabic character recognition systems have been proposed since the last three decades. Many systems reported high recognition rates or high speed rate, however, they overlooked one of them but not both. Any real-time system should have both factors. In this paper, a high-performance Arabic character recognition system is introduced. The goal of the system is to maximize accuracy and minimize speed. The goal has been achieved through developing a high-accuracy yet simple Arabic character recognition technique; the technique is optimized using OpenMP-like technique for utilizing multithreads or multi-processors.**

1 INTRODUCTION

Since the advent of computers, search for better interaction between man and computer has been going on. The goal has been always to provide the computer with information that can be retrieved and utilized afterwards. The merit of such interaction is usually evaluated through answering two questions. The first is how accurate the process of transferring the information to the computer is. The second is will the information be processed within the time specified by the user. The answer to the first question involves researching the area of speech and character recognition. The goal is to find the most accurate way to feed the machine with human voices or scripts. The second question involves finding the optimum approach to attain that goal in terms of meeting the user's time requirement. The emergence of multi-processing machines provided additional frontiers to meet such requirements. Hence, an integration of a speech or character recognition system into a parallel environment should be very promising.

Throughout the years, researchers have put great effort in Latin and other character recognition as a way of facilitating man-machine interface [1][2][3][4]. Such research has led to the availability of many commercial optical character recognition systems for Latin scripts. Similarly, since the early eighties, a number of Arabic character recognition systems have been proposed [5][6][7][8][9][10]. Most systems achieved high recognition rates which is the rate of the number of recognized characters to the total number of characters. Despite these high rates, most current systems ignored the speed factor that has the same importance as the recognition rate. Building a recognition system with a 100% recognition rate would be useless if the recognition speed fails in matching the required specifications of real-time applications. As a result, our focus should be on building a recognition system that achieves both full accuracy and high speed leading to an optimal communication interface between the system and its users.

2 FEATURES OF ARABIC CHARACTERS

Arabic characters possess some unique features that make them different from English especially for the recognition problem. The following are the main characteristics of Arabic characters [11]:

1. The Arabic language consists of 28 main characters. Whereas English characters can appear in two forms (upper and lower), Arabic characters can have four different shapes, depending on the position of the character within the word (beginning, middle, end, and isolated).

2. The Arabic word is always cursive.
3. The Arabic text is written from right to left.
4. Many Arabic characters (15 out of 28) have dots which are positioned at a suitable distance above, below, or within the letter body. Dots can be single, double, or triple. Different Arabic letters can have the same body and differ in the number or position of dots identifying them. Letters Baa, Taa, and Thaa are examples of such a case.

   In Addition to the previous points we add:

5. Although The Arabic language consists of 28 main characters there are many more cases for some characters like 'لا' which is lam 'ل' followed by Alif 'ا' but written in a different form ('لا' instead of 'لـا').
6. As well as there are different cases of dots there are Hamza 'ء' Maed 'آ' .

7. Having 28 characters in the Arabic alphabet; each character has two to four different forms that depend on its position in the word or sub-words and after adding other cases, as Hamza, there are 121 classes to be recognized. The Arabic character set is shown in Fig. 1 [5].

8. An Arabic word can have one or more sub-words, e.g., there are three sub-words in the word shown in Fig. 2.

9. We have also noticed that Arabic words may horizontally overlap and characters may stack on others as shown in Fig. 2 for Raa 'ر' and Hha 'ح'.

No	Name	separated	starting	middle	ending
1	Alif	ا	ا	ـا	ـا
2	Baa	ب	بـ	ـبـ	ـب
3	Taa	ت	تـ	ـتـ	ـت
4	Tha	ث	ثـ	ـثـ	ـث
5	Jeem	ج	جـ	ـجـ	ـج
6	Hha	ح	حـ	ـحـ	ـح
7	Kha	خ	خـ	ـخـ	ـخ
8	Dal	د	د	ـد	ـد
9	Thal	ذ	ذ	ـذ	ـذ
10	Raa	ر	ر	ـر	ـر
11	Zay	ز	ز	ـز	ـز
12	Seen	س	سـ	ـسـ	ـس
13	Sheen	ش	شـ	ـشـ	ـش
14	Sad	ص	صـ	ـصـ	ـص
15	Dahd	ض	ضـ	ـضـ	ـض
16	Tta	ط	طـ	ـطـ	ـط
17	Tha	ظ	ظـ	ـظـ	ـظ
18	Ain	ع	عـ	ـعـ	ـع
19	Ghain	غ	غـ	ـغـ	ـغ
20	Faa	ف	فـ	ـفـ	ـف
21	kaaf	ق	قـ	ـقـ	ـق
22	kaf	ك	كـ	ـكـ	ـك
23	Lam	ل	لـ	ـلـ	ـل
24	Meen	م	مـ	ـمـ	ـم

25	Noon	ن	نـ	ـنـ	ـن
26	Haa	ه	هـ	ـهـ	ـه
27	Waw	و	و	ـو	ـو
28	Yaa	ي	يـ	ـيـ	ـي
More Cases					
29		إ	إ	ـإ	ـإ
30		أ	أ	ـأ	ـأ
31		آ	آ	ـآ	ـآ
32		لا	لا	ـلا	ـلا
33		لأ	لأ	ـلأ	ـلأ
34		لآ	لآ	ـلآ	ـلآ
35		ئ	ئـ	ـئـ	ـئ
36		ء			ء
37		ؤ	ؤ	ـؤ	ـؤ
38		ى		ـى	ـى

Figure (1): Arabic alphabet in all its forms.

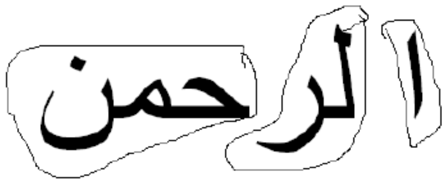


Figure (2): Arabic word consists of three sub-words and overlapping between the second and the third sub-word.

3 Moments

Region moment representation interprets a normalized gray level image as a probability density function of a 2D random variable. Properties of this random variable can be described using statistical characteristics - moments. A moment of order (p+q) is dependent on scaling, translation, rotation, and even on gray level transformations and is given by

$$m_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} x^p y^p g(x,y)\,dxdy$$

The central moment

$$\mu_{pq} = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} (x-x_c)^p (y-y_c)^q g(x,y)\,dxdy$$

Let $f(x,y) = (x-x_c)^p (y-y_c)^q g(x,y)$

We use composite trapezoidal rule for evaluating the double integral

$$I_T = \int_0^M \int_0^N f(x,y)\,dxdy$$

$$I_T = \frac{hk}{4}\Big[\{f_{00} + f_{0M} + 2(f_{01} + f_{02} + \cdots + f_{0,M-1})\}$$
$$+ 2\sum_{i=1}^{N-1}\{f_{i0} + f_{iM} + 2(f_{i1} + f_{i2} + \cdots + f_{i,M-1})\}$$
$$+ \{f_{N0} + f_{NM} + 2(f_{N1} + f_{N2} + \cdots + f_{N,M-1})\}\Big]$$

Where M and N image size, x_c , y_c are the co-ordinate of the region's centriod , $h = k = 1$ and

$x_c = \frac{m_{10}}{m_{00}}$, $y_c = \frac{m_{10}}{m_{00}}$

The normalized un-scaled central moments

$\vartheta_{pq} = \frac{\mu_{pq}}{(\mu_{00})^{\lambda}}$, where $\lambda = \frac{p+q}{2} + 1$

A less general form of invariance is given by seven invariant moments characterizing: rotation, translation, and scale [31].

$$\emptyset_1 = \vartheta_{20} + \vartheta_{02}$$
$$\emptyset_2 = (\vartheta_{20} - \vartheta_{02})^2 + 4\vartheta_{11}^2$$
$$\emptyset_3 = (\vartheta_{20} - 3\vartheta_{i2})^2 + (3\vartheta_{21} - \vartheta_{03})^2$$
$$\emptyset_4 = (\vartheta_{30} - \vartheta_{i2})^2 + (\vartheta_{21} - \vartheta_{03})^2$$
$$\emptyset_5 = (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{30} + \vartheta_{12})[(\vartheta_{30} - \vartheta_{12})^2 - 3(\vartheta_{21} - \vartheta_{03})^2] + (3\vartheta_{21} - 3\vartheta_{03})(\vartheta_{21} + \vartheta_{03})[3(\vartheta_{30} - \vartheta_{21})^2 - (\vartheta_{21} + \vartheta_{03})^2]$$
$$\emptyset_6 = (\vartheta_{20} - \vartheta_{02})[(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2] + 4\vartheta_{11}(\vartheta_{30} + \vartheta_{12})(\vartheta_{21} + \vartheta_{03})$$
$$\emptyset_7 = (3\vartheta_{21} - \vartheta_{03})(\vartheta_{30} + \vartheta_{12})[(\vartheta_{30} + \vartheta_{12})^2 - 3(\vartheta_{21} + \vartheta_{03})^2] - (\vartheta_{30} - 3\vartheta_{12})(\vartheta_{21} + \vartheta_{03})[3(\vartheta_{30} + \vartheta_{12})^2 - (\vartheta_{21} + \vartheta_{03})^2]$$

4 PROPOSED TECHNIQUE

Arabic character recognition requires several steps, mainly: segmentation, feature extraction and identification. In this paper, the first step is achieved by using a local diffusive segmentation method. There exist a wide variety of ways to achieve segmentation; however, it is not the subject of this paper. All contiguous pixels, which share a given point-based characteristic of the object, or are surrounded by those that do, are considered as object pixels and those outside the included region, are considered as background. The result is a group of contiguous pixels, which collectively represent the object. The boundary pixels of the object are then extracted from the segmented object pixels by a simple iterative trace, around the outside of the object that continues until the starting point is reached.

In the second stage, the input is a set of character images. First they must be transformed into a negative state – white object on black background. Second moment's invariants are calculated for each image to be used as key features to identify character images as shown in the appendix at the end of the paper. By definition an image moment is a certain particular weighted average (moment) of the image pixels' intensities, or a function of such moments, usually chosen to have some attractive property or interpretation. We used them because they are constant to an image even if it is scaled or rotated. Moments had been used before for image retrieval and identification [12][13] in database systems, which we will use in this research for OCR.

In the third stage, after explaining the process of feature extraction for Arabic characters classification, we enhance the recognition capability of the system with a neural network to classify the characters. We used a multi-layer perceptron (MLP) neural network to equip the proposed engine with learning the features of the different characters, so as, the system recognition capability would be enhanced in this way.

The input pattern of the network consists of the feature vector composed of the seven moments calculated for each character. All the input pattern's features have been passed to 121 output nodes, feeding 121 possible input cases to train the network.

5 Experimental Results

In this section we will present and evaluate the experimental results of our technique. We used a random set of characters – as shown in figure 3 - to evaluate our technique and run multiple tests on multiple subsets and then calculate the Mean Absolute Error for each test. Our technique has a Mean Absolute Error =0.02.

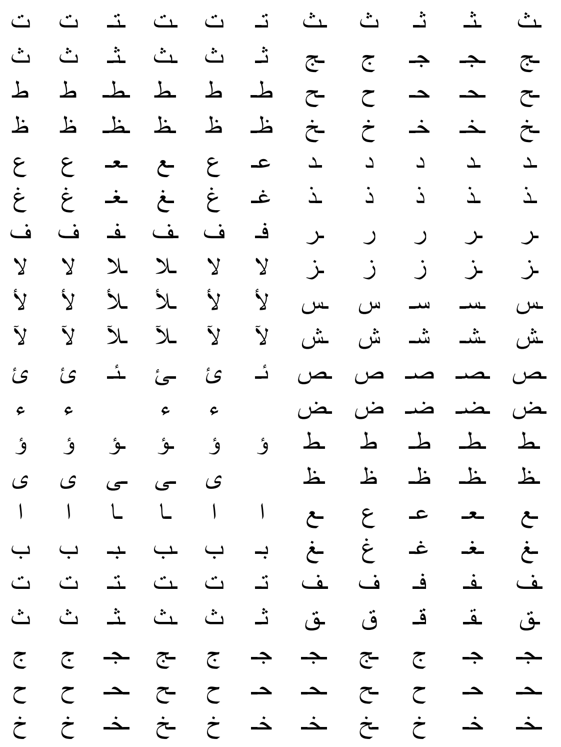


Figure (3): An example of samples used in the experimental test.

6. Conclusion

This paper presents a new technique for Arabic OCR CBIR based on moment's invariants and neural networks. The proposed method consists of two parts: feature extraction and character identification. The experimental results show that our technique is quite promising. We also considered a larger set of 121 Arabic character cases instead of the classic 100 one.

We will work in the future on the Tashkil signs of the Arabic characters. The small signs that change the sound of the word. Moreover, we might consider other fonts and formats. Finally, we would like to explore other types of NNs that might enhance and speed the recognition process.

References

[1] Mantas, J.. "An overview of character recognition methodologies". *Pattern Recognition* 19 (6), pp. 425-430, 1986.

[2] 2 Mori, S., Suen, C., Yamamoto, K.,. "Historical review of OCR research and development". *Proceedings of the IEEE* vol. 80, no. 7, pp. 386-405, 1992

[3] Nagy, G., Seth, S.. *Modern optical character recognition*. Froehlich, F., Kent, A., Hall, C., (Eds.), The Froehlich/Kent Encyclopedia of Telecommunications. Vol. 11, Marcel Decker, New York, pp. 473-531. 1996

[4] Pal, Roy , Tripathy and Llados, Multi-oriented Bangla and Devnagari text recognition, Pattern Recognition, vol. 43, pp. 4124–4136, 2010

[5] Abdelazim, H., Mousa, A., Salih, Y., Hashish, M.,. Arabic text recognition using a partial observation approach. In: *Proceedings of the 12th National Computer Conference*. Riyadh, Saudi Arabia, pp. 427-437, 1990

[6] Al-Badar, B., Mahmoud, S.. Survey and bibliography of Arabic optical text recognition. *Journal of Signal Processing*, vol. 41, no 1, pp. 49-77, 1995

[7] Alherbish, J.,. *High Performance Arabic Character Recognition*. Ph.D. Thesis, University of Connecticut, CT, 1996

[8] Khellah, F., Mahumod, A.,. “Recognition of hexagonally sampled printed Arabic characters”. The *Arabian Journal for Science and Engineering*, vol. 19, no 4A, pp. 565-586, 1994

[9] A Amin,H Al-Sadouni and S Fischer, “Hand-Printed Arabic Character Recognition System Using An Artificial Network”, *Pattern Recognition*, vol. 29, no. 4, pp. 663 675, 1996

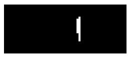
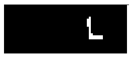
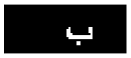
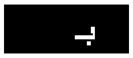
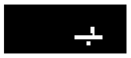
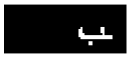
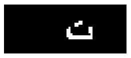
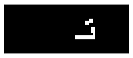
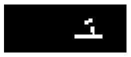
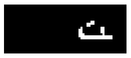
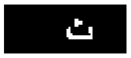
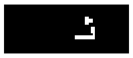
[10] Alherbish, Ammar, “High-performance Arabic character recognition”, *The Journal of Systems and Software*, vol. 44, pp. 53-71, 1998

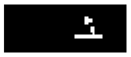
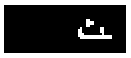
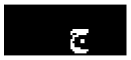
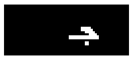
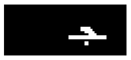
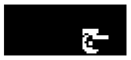
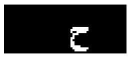
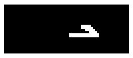
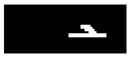
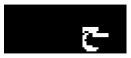
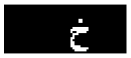
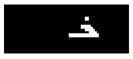
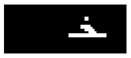
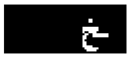
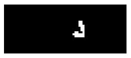
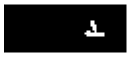
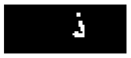
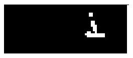
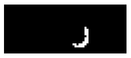
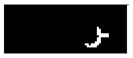
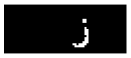
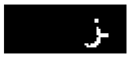
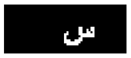
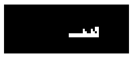
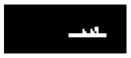
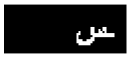
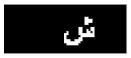
[11] Mahmoud, S., Abu Haiba, I., Green, R., 1991. “Skeletonization of Arabic characters using clustering based skeletonization algorithm (CBSA)”. *Pattern Recognition* 24 (5), 453±464.

[12] Rizon and others, “Object Detection using Geometric Invariant Moment”, *American Journal of Applied Sciences* vol. 2, no. 6, pp. 1876-1878, 2006

[13] Amr, Amin, El-Kafrawyy and Sauber, Using Statistical Moment Invariants and Entropy in Image Retrieval , *Proceeding of IJCSIS* August 2009

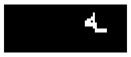
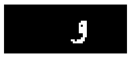
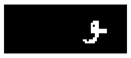
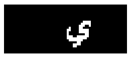
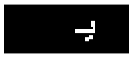
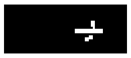
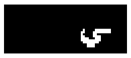
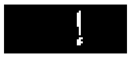
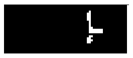
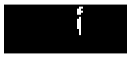
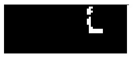
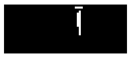
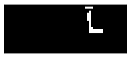
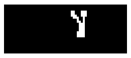
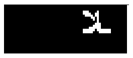
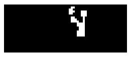
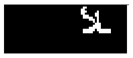
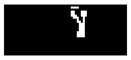
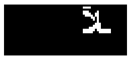
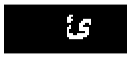
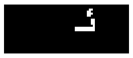
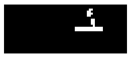
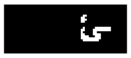
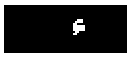
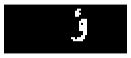
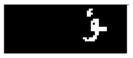
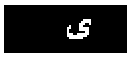
Appendix: The images of all possible shapes of characters and the corresponding moment's invariants values

No.	ID	Name	Phi1	Phi2	Phi3	Phi4	Phi5	Phi6	Phi7
1	1		0.51348593	0.24046356	0.19952337	0.19374598	0.034041221	-0.04041018	0.002213509
2	4		0.57243076	0.17605628	1.82876575	1.27594321	0.239500325	-0.02237976	-0.49475388
3	5		0.48383499	0.0992986	1.4081342	0.13823916	0.047961495	-0.01405392	0.064688886
4	6		0.39018496	0.03360662	0.04917335	0.44484608	-0.17132452	0.000184426	-0.02308836
5	7		0.39663514	0.05949592	0.10159821	0.14555496	-0.01826053	0.003426853	-0.00401176
6	8		0.5320821	0.16589933	0.57756122	0.04493097	0.000736081	-0.00814528	0.001427151
7	9		0.47004304	0.04401122	0.36397648	0.13113947	-0.00112885	-0.00070796	-0.00547026
8	10		0.49996501	0.03603016	2.43119929	1.19793445	0.653559805	-0.00728851	0.199812673
9	11		0.47286402	0.00406314	3.97124839	1.27976774	0.556503944	-0.00019471	0.058840111
10	12		0.51666705	0.1004805	1.02292102	0.29373638	-0.04422433	0.002262942	0.00416058
11	13		0.45339525	0.01851946	0.89272325	0.31749632	-0.04154231	0.00013347	-0.00252657
12	14		0.50830257	0.04220102	2.88754321	1.07890468	0.583543175	-0.00364621	0.117642868

13	15		0.48274467	0.00106848	4.81410931	1.5048024	0.71629581	6.96E-05	0.098339254
14	16		0.49914352	0.06567965	1.79333051	0.62278344	-0.25261268	0.003766646	0.02671461
15	17		0.39243384	0.03418779	0.05740282	0.04769106	-0.00126703	0.000422797	0.005619581
16	18		0.31487832	0.02968625	0.22544119	0.24341286	-0.01171848	0.00480792	0.005218898
17	19		0.37507133	0.06760058	0.52443502	0.05478958	0.000941061	-0.00141765	0.004488728
18	20		0.33738497	0.00428418	0.36012477	0.29056537	-0.01135912	-8.39E-05	-0.0048217
19	21		0.44713421	0.04830117	0.10687816	0.0802836	-0.01054278	0.002336741	0.02546698
20	22		0.339376	0.05199642	0.69560542	0.33154382	0.026413353	0.010152174	-0.00358594
21	23		0.40914962	0.09933534	1.21777452	0.15872775	-0.00474901	-0.00463757	0.024434066
22	24		0.3762731	0.00495493	0.46773178	0.39574875	-0.01218962	-0.00031453	0.01173723
23	25		0.47351682	0.07617086	0.01647283	0.0354496	-0.01100364	0.004934931	0.013189353
24	26		0.35008637	0.02039904	1.72519727	0.62194335	0.031632725	0.003624865	-0.1187025
25	27		0.40571817	0.05282934	2.53407073	0.50559343	-0.0199701	-0.00033337	0.006078222
26	28		0.38767279	0.00944428	0.15425486	0.26209005	-0.03257097	-0.00012155	-0.00130883
27	29		0.298368	0.02307172	0.24485422	0.11112118	0.008144392	-0.00036986	0.002401827
28	32		0.30781646	0.00317893	0.81127268	0.2610676	-0.00885782	6.98E-06	-0.00140375
29	33		0.47734634	0.11832847	2.05680032	1.48761503	2.066684681	-0.11568866	0.478923171
30	36		0.43642035	0.04344279	3.23003476	1.90416966	2.192387387	-0.02162654	-0.26580303
31	37		0.48133819	0.10655886	1.26044734	0.70784518	0.194330509	-0.007659	0.079123251
32	40		0.42814981	0.07913416	0.57383813	0.06647803	-0.02123773	-0.00118123	-0.03304087
33	41		0.70118519	0.26523992	7.61230905	4.67795508	20.01433056	-0.66531308	7.769188646
34	44		0.53659426	0.07762224	3.5515309	1.564404	1.408524589	-0.04938937	1.201390156
35	45		0.45096737	0.10817553	0.19475385	0.01059584	0.001513537	0.000617967	-5.67E-05
36	46		0.4977583	0.19866177	0.15679503	0.48068278	-0.38145188	0.055757195	-0.03105472
37	47		0.54903994	0.25260348	0.46485532	0.42298094	-0.07388397	0.070917507	0.007285662
38	48		0.49811925	0.14954637	0.3425857	0.0546775	0.003226128	-0.00110511	0.000566202
39	49		0.47165831	0.08629934	0.88766991	0.57568256	-0.00016377	-0.00283923	-0.06329362

40	50	شـ	0.45561101	0.04063789	2.48190954	1.35944194	0.061703296	0.005712444	-0.44541166
41	51	ـشـ	0.47146677	0.04893759	3.96328692	1.18016784	-0.06570969	0.00988628	-0.16107619
42	52	ـش	0.47313119	0.06982351	2.34983907	0.56156316	-0.03108245	0.002910016	-0.06416703
43	53	ص	0.52584543	0.1952889	0.383267	0.1068067	0.010834776	0.035197531	-0.02077468
44	54	صـ	0.53751895	0.23005276	0.30881443	1.76313794	-4.18845667	0.285195839	-0.17980211
45	55	ـصـ	0.56511615	0.25703085	0.912166	1.63001507	-1.48906731	0.293787954	0.082297923
46	56	ـص	0.56342995	0.23133659	1.04093806	0.07042356	-0.03052889	0.011933154	0.00123734
47	57	ض	0.5052542	0.16093846	0.71670525	0.27558363	0.05523719	0.036673994	-0.10029915
48	58	ضـ	0.50648777	0.16888047	0.8653345	1.76814818	-2.22536779	0.19646211	-0.66414349
49	59	ـضـ	0.53190065	0.19287442	1.77957362	1.54332944	-0.14589987	0.185996553	-0.19937968
50	60	ـض	0.53590809	0.18603294	1.73342261	0.2188929	-0.08172129	0.012110217	0.014225763
51	61	ط	0.31723088	0.00257222	0.93247919	0.43341874	0.071365529	0.000205311	0.037471609
52	62	طـ	0.36908746	0.00696639	2.49239371	0.70032427	0.07943616	0.001162136	-0.17237329
53	63	ـطـ	0.39243118	0.0191143	3.43223572	0.80355251	0.010399086	0.001677359	-0.04954194
54	64	ـط	0.34595044	0.01261367	1.57722631	0.72412425	-0.09036928	-0.00010075	-0.01300902
55	65	ظ	0.30708253	0.00173904	0.68304714	0.27041266	0.031970052	1.75E-05	0.017567064
56	66	ظـ	0.35847087	0.00670283	2.12596731	0.61816948	0.038821146	0.000717733	-0.18917341
57	67	ـظـ	0.37688984	0.01204596	3.21181655	0.73262588	-0.05017632	0.001197258	-0.05447011
58	68	ـظ	0.33130565	0.00680178	1.41032828	0.52942637	-0.03747902	-1.12E-05	-0.00317531
59	69	ع	0.45685811	0.09245099	0.05662764	0.00338339	0.027475879	-0.00178365	-0.00421877
60	70	عـ	0.31959398	0.04788684	0.33289061	0.14204883	0.004217062	0.003644904	-0.00457358
61	71	ـعـ	0.33216485	0.05028158	0.90614155	0.14151876	-0.01166794	-0.00150921	0.013726506
62	72	ـع	0.34685118	0.01337728	0.03925785	0.0587916	-0.00391578	-0.00011965	-0.00023357
63	73	غ	0.49850264	0.12968023	0.3951615	0.08261349	0.0213848	0.008592961	0.014761955
64	74	غـ	0.31984	0.02571269	0.79043559	0.29345921	0.005760083	0.001672272	-0.02907277
65	75	ـغـ	0.34323666	0.01845286	2.02388681	0.47843874	-0.01331572	0.000214616	-0.00146311
66	76	ـغ	0.37648466	0.02792528	0.11151113	0.06262548	-0.00453087	0.000227474	0.000530145

67	77	ف	0.55929609	0.13494754	0.05752542	0.76270188	-0.7741162	-0.01068872	-0.17644214
68	78	فـ	0.35714453	0.02988435	1.3549732	0.60498335	0.085625807	-0.0007495	-0.04086737
69	79	ـفـ	0.35863162	0.00259689	2.98127304	1.08437061	0.470450873	-0.00035284	0.065483644
70	80	ـف	0.56985751	0.16241161	0.48246512	0.84136843	-0.56749662	0.045175184	-0.26687203
71	81	ق	0.39917586	0.02570591	1.42277052	0.95286162	0.544585089	-0.00698384	0.324005379
72	82	قـ	0.39104999	0.0454164	1.71256167	0.78419558	0.243237758	-0.00255553	0.039398567
73	83	ـقـ	0.39943194	0.00788589	4.26460022	1.69460229	1.714938137	-0.00234839	0.375666608
74	84	ـق	0.36946601	0.00857327	1.27255674	0.57488232	0.250879864	-0.00145953	0.105600887
75	85	ك	0.33923544	0.02333804	0.30236607	0.39889992	0.002399449	-0.00011294	0.026843426
76	86	كـ	0.35629151	0.01408028	1.17887974	0.40679687	0.015252973	0.000149787	-0.01004725
77	87	ـكـ	0.37425764	0.00027277	2.72885571	0.63961861	-0.01323291	-6.37E-07	-4.43E-05
78	88	ـك	0.34349135	0.00732496	1.21230853	0.51238548	0.054450359	-6.81E-05	0.001514839
79	89	ل	0.51337681	0.08096054	1.18824525	1.44102333	0.926792124	-0.03753039	0.805073971
80	90	لـ	0.54594989	0.12449657	3.03552476	1.72194656	1.274361162	-0.02957832	0.524015985
81	91	ـلـ	0.47975857	0.01833214	5.3198	2.14456651	2.138104606	-0.00406563	0.374599085
82	92	ـل	0.45077782	0.02548306	1.99331559	1.2388748	1.119310184	-0.00372299	0.320312812
83	93	م	0.35243232	0.06291575	0.28380145	0.25965215	0.046745385	-0.00783806	0.012915806
84	94	مـ	0.35510974	0.08702415	0.12928189	0.37985199	-0.15940024	0.019730069	-0.01317406
85	95	ـمـ	0.40354996	0.12088528	0.43885191	0.21375456	0.007251905	0.013338198	0.010907323
86	96	ـم	0.34526509	0.01685253	0.21969492	0.41984	-0.02922756	-5.89E-05	0.09165811
87	97	ن	0.42798747	0.0102825	0.60618606	0.56021329	0.228720073	-0.0022588	0.197541817
88	98	نـ	0.48524722	0.05615896	1.341085	1.14812336	0.044927575	-0.00939169	0.101396996
89	99	ـنـ	0.46110015	0.03455434	2.58393153	0.96069033	0.186244816	0.00262184	-0.05794815
90	100	ـن	0.40307272	0.01082667	0.36302466	0.1966122	0.034902248	-0.00107058	0.027821502
91	101	ه	0.19994213	0.01083269	0.01098637	0.00385729	9.63E-06	-9.03E-06	1.32E-06
92	102	هـ	0.26418737	0.01792356	0.55851212	0.27214789	0.007875686	0.001694131	-0.02159765
93	103	ـهـ	0.28585633	0.01895186	0.04437821	0.00497271	-8.98E-06	2.53E-06	-2.53E-06

94	104		0.35555475	0.04968204	0.69779023	0.13506525	0.098546585	0.001202368	0.044208079
95	105		0.36548155	0.06510147	0.24379733	0.01809119	-0.00676455	0.001145881	-0.00681385
96	108		0.32023733	0.0216649	0.29916941	0.07054431	0.013216415	-0.00092237	-0.00844875
97	109		0.31869661	0.0221274	0.05130322	0.07888822	0.004382359	-3.33E-05	0.002122836
98	110		0.37065889	0.01008951	0.18064172	0.3948456	-0.09358552	-6.30E-05	-0.0123305
99	111		0.36771605	0.03011438	0.07322082	0.07878852	-0.01206227	0.000875763	-0.00082109
100	112		0.33554597	0.03674416	0.91106326	0.11783776	-0.0175256	-0.00200942	0.046881679
101	113		0.97535532	0.90601136	0.51647278	0.5750439	0.328992068	-0.57553804	0.002479303
102	116		0.58539474	0.13962764	1.2427269	1.29063157	0.27977799	-0.03312036	0.160666221
103	117		0.693844	0.43928626	0.08203627	0.05764718	0.001131213	-0.00358039	0.002916343
104	120		0.66739015	0.26005484	3.86282133	2.2209418	2.243294287	-0.10191519	-1.54906706
105	121		0.66146133	0.39274044	0.09686717	0.05084986	0.001361809	-0.00348406	0.000776105
106	124		0.71282959	0.31954074	3.80166795	2.52469717	3.315030946	-0.20171047	-1.9686574
107	125		0.4058173	0.06851054	0.71580718	0.17819819	-1.66E-05	0.001023337	0.001670091
108	128		0.4168099	0.02500434	1.0744891	0.46167945	-0.01831265	-3.09E-06	-0.00263816
109	129		0.41510455	0.0611749	0.76074122	0.11390652	0.058997576	0.000297602	-0.01066326
110	132		0.42900771	0.04915118	0.52841671	0.39069815	-0.0025307	4.36E-05	-0.01435372
111	133		0.43430483	0.09143368	0.65452084	0.13986546	0.013127801	0.002637119	0.005410006
112	136		0.44419649	0.04304991	1.25306357	0.6783396	-0.00832477	-0.00020224	-0.01322404
113	137		0.35594139	0.00102107	0.25396319	0.0020656	0.043081936	-0.0002108	-0.01010137
114	138		0.53017833	0.05544939	3.51325218	1.76024521	1.522847666	-0.01761352	0.507587697
115	139		0.49060856	0.00162721	5.36433263	1.85974736	1.416365603	-0.00080826	0.207746645
116	140		0.44582019	0.0120774	0.75382227	0.61228701	-0.0048871	-0.00507747	-0.69142341
117	141		0.20496123	0.00347482	0.02203227	0.00583988	-7.40E-06	-1.11E-07	-1.09E-07
118	145		0.50049732	0.13109899	0.72572679	0.44994704	-0.15284186	0.019326743	0.460753464
119	148		0.42615138	0.04580369	0.90284972	0.6904555	-0.1063073	0.006382922	0.469637319
120	149		0.33429005	0.02627507	0.01902429	0.19118902	-0.03817926	-0.00028624	-0.00812037

121	152	[redacted]	0.38308439	0.06758365	0.55861758	0.05462001	0.00310965	-0.00215921	0.015516476

AL-IMAM: A Comprehensive Database for Arabic Text Mining

Ibrahim F. Imam & Ahmed Abd-Allah

Computer Science Department
Arab Academy for Science, Technology and Maritime Transport
Cairo, Egypt
ifi05@yahoo.com

***Abstract*— Arabic language is one of the most sophisticated languages in the world. There are many efforts to introduce language resourceful systems for text mining in the Arabic language. This paper presents a novel database and a framework to support applications in Arabic text mining. The database contains extensive information of more than ten millions diacritic Arabic words and is linked with the English WordNet and SUMO. This database is designed to support applications such as translation, categorization, similarity and opinion mining. The paper presents an analysis to show the speed and the kind of information that can be retrieved.**

1. INTRODUCTION

The Arabic language is very rich language. It permits expandability in different ways. Due to the development slowdown in the Arabic countries in the past centuries, new terminologies appeared in the western countries that has no substitution in Arabic. Globalization accelerated transferring these terminologies to the Arabic world.

Research depends on rule-based approaches for extracting information from text. Rule-based approaches analyze the grammar of each sentence and match it with predefined rule. A correct match identifies the type of each token. More rules are needed to analyze tokens [6]. Usually, this is a very complex and time consuming process. Researchers evade the rule-based approaches and exploit statistical approaches [4] [5]. Statistical approaches tend to estimate the correct matches based on the probability of tokens, phrases, sentences, etc. A new approach has emerged where necessary information are stored in database and retrieved much faster than the rule-based approaches and more accurately than the statistical approaches. The only disadvantage of this approach is that it requires massive storage.

Natural language processing and text mining applications require rich corpus to accomplish the given task. In this paper, a comprehensive database is created to cover all possible information about the Arabic words and their relation to English words. There are many issues to be taken into consideration when creating such database. Among these issues, the sense of the word, the search index, word categorizations, word ambiguity, etc. English language and implemented the Word Net system. Unfortunately, the Arabic language is more complicated than English. Also, the Arabic version of the word Net is far from complete. This is because the Arabic language is evolving faster than any other language and it is easy to migrate terms from other languages to the Arabic language [9] [10]. Therefore, it was necessarily to implement a gigantic database that captures as much information as possible about all Arabic words.

The proposed database is very helpful in Arabic information retrieval and text mining. Millions of Arabic words handled in it. The huge amount of data help us to improve output of statistical text mining approaches [and we can save our time and effort consumed in rule based systems. This huge database can help to improve performance of many text mining applicatiox like information retrieval as proposed in [1] & [7], machine translation [8] & [11], text summarization and text similarity [3] and can used to improve text categorization [2].

2. DATABASE DESIGN

Designing the proposed database was a challenge. This research investigated four different designs. The initial design started by collecting Arabic documents using internet crawler. The text in the collected documents is tokenized and each token is translated by a free source translator. Also, a part of speech tagging is assigned to each token. The second design started by generating all possible n-gram combinations of the 28 Arabic alphabets. To ensure that a token is a word, we searched over the internet for each token. In the third design, the database initiated using an Arabic-English online dictionary. All these designs failed when attempting to assign relationships among the words. Also, searching for the diacritic corresponding was extremely difficult process. The successful design started by collecting all possible diacritic words. Even though, this approach seemed illogical, however, the design was much simpler and successful.

Data Collection and Maintenance*:* The Arabic spoken citizens originate new words in many ways. One way to originate new word is by writing the phonetic of a foreign word in Arabic letters. For example, the English word "Internet" may be written in Arabic "*إنترنت*". Such words have to be added to the database and linked with all other entities. Figure 1 shows an abstract entity relationship diagram of the database. The main entity is a list of all diacritic words and a key for each one. Figure 2 shows an example of one word "الفِلاحَة".

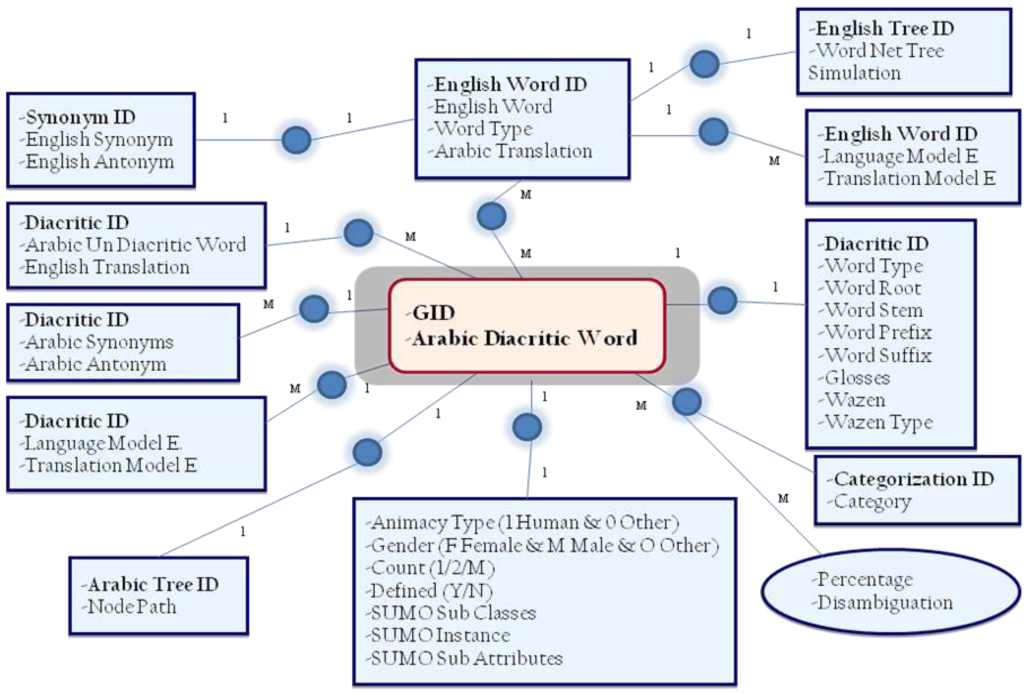
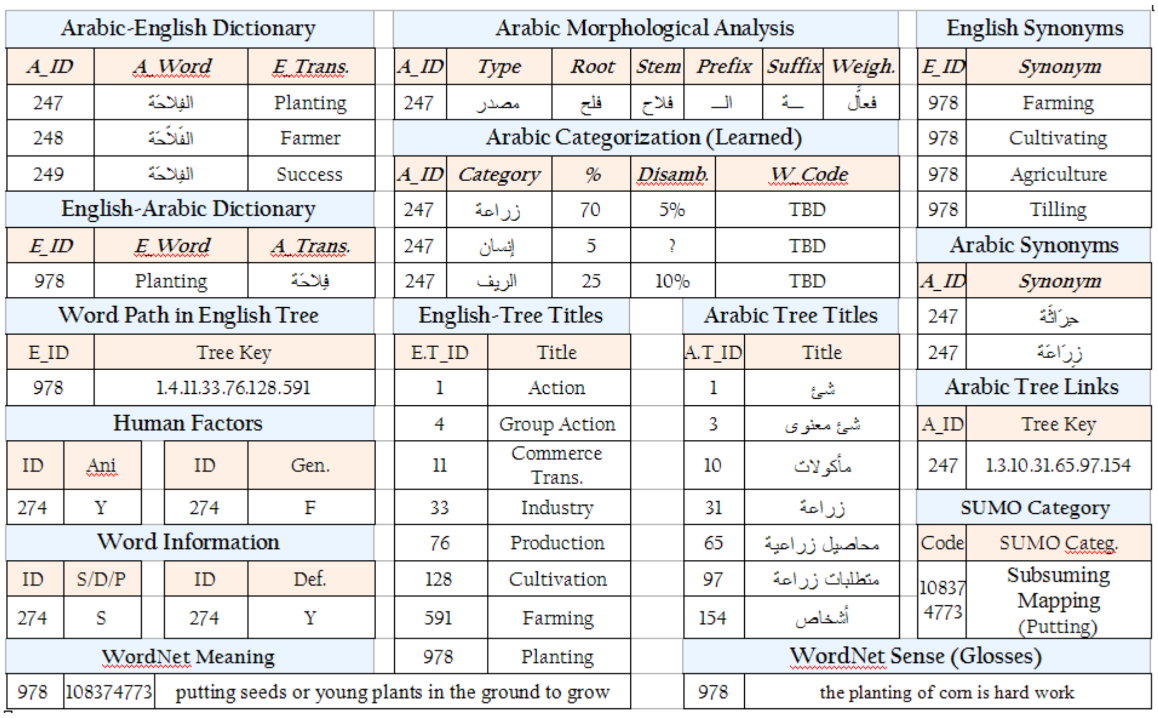


Figure 1: An Entity Relationship Diagram of Al-Imam Database.



Arabic-English Dictionary

A_ID	*A_Word*	*E_Trans.*
247	الفِلاحَة	Planting
248	الفَلّاحَة	Farmer
249	الفَلاحَة	Success

English-Arabic Dictionary

E_ID	*E_Word*	*A_Trans.*
978	Planting	فِلاحَة

Word Path in English Tree

E_ID	Tree Key
978	1.4.11.33.76.128.591

Human Factors

ID	Ani	ID	Gen.
274	Y	274	F

Word Information

ID	S/D/P	ID	Def.
274	S	274	Y

WordNet Meaning

978	108374773	putting seeds or young plants in the ground to grow

Arabic Morphological Analysis

A_ID	*Type*	*Root*	*Stem*	*Prefix*	*Suffix*	*Weigh.*
247	مصدر	فلح	فلاح	الـ	ـة	فِعال

Arabic Categorization (Learned)

A_ID	*Category*	*%*	*Disamb.*	*W_Code*
247	زراعة	70	5%	TBD
247	إنسان	5	?	TBD
247	الريف	25	10%	TBD

English-Tree Titles

E.T_ID	Title
1	Action
4	Group Action
11	Commerce Trans.
33	Industry
76	Production
128	Cultivation
591	Farming
978	Planting

Arabic Tree Titles

A.T_ID	Title
1	شيء
3	شيء معنوى
10	مأكولات
31	زراعة
65	محاصيل زراعية
97	متطلبات زراعة
154	أشخاص

WordNet Sense (Glosses)

978	the planting of corn is hard work

English Synonyms

E_ID	*Synonym*
978	Farming
978	Cultivating
978	Agriculture
978	Tilling

Arabic Synonyms

A_ID	*Synonym*
247	حِرَاثَة
247	زِرَاعَة

Arabic Tree Links

A_ID	Tree Key
247	1.3.10.31.65.97.154

SUMO Category

Code	SUMO Categ.
108374773	Subsuming Mapping (Putting)

Figure 2: An example of Al-Imam Database.

3. Database components and statistics

Arabic-English Dictionary: Contains Arabic diacritic words and their equivalent English translation according to diacritic word part of speech. It contains 3588192 Diacritic words generating about 17000000 Arabic English translation entries. This table contains 1644508 nouns and 1938817 verbs.

Arabic Morphological Analysis: Contains morphology of diacritic Arabic words. morphology that we can get complete analysis of each word, we can get root of the word, Stem of the word which is the smallest component of the word, all word prefixes, antefix, suffixes and postfixes. Word (ليكاتبنهم) has (لـ) as its antefix, (يـ) as its prefix,(ن) as its suffix,(هم)as its postfix, (كتب) as its root,(كاتب) as its stem and (فاعل) as its weight. This component contains 11041256 Arabic diacritic words and their morphological analysis (word type, root, stem, prefix, suffix and weight).database it contains 6646058 nouns, 4394339 verbs and 859 stop words.

Arabic Synonyms: Synonyms are different words with similar meanings (زراعه & حراثه) are synonyms of (فلاحه). Synonym words called synonymous, and the state of being a synonym is called synonymy. This component contains 197895 Arabic diacritic words and their Arabic diacritic synonyms. It contains 55372 nouns, 64192 verbs, 38263 adjectives, 39887 masdar and 181 stop words. English Arabic Dictionary: Contains 255752 English words and their Arabic translation according to English word part of speech. It contains 116314 nouns, 89457 verbs, 47919 adjectives, 1748 adverb and 314 stop words. English Synonyms: Contains 199880 English words and their English synonyms. It contains 116236 nouns, 45479 verbs and 38165 adjectives. English Antonyms: words with opposite or nearly opposite meanings like(short and tall Contains 37107 English words and their English antonyms. It contains 9049 nouns, 10881 verbs and 17177 adjectives. Word Path in English Tree: Contains 94841 English word and their English word net trees. It contains 81856 nouns and 12985 verbs. Arabic Tree Links: Contains 9756 Arabic word net tree paths. It contains 6404 nouns, 2500 verbs and 852 adjective.

Experiments

The goal of this experiment is view how far the proposed database is rich to help in Arabic text mining area. Using many Arabic articles in different areas we found that proposed database cover about 90% of words morphology, 90% of words translation, 30% of Arabic words synonyms and 20% of words Tree.

Table 1: Experimental results.

Total Words	Translation	Morphology	WordNet Tree	ARSynonyms
134	0.96	0.94	0.19	0.33
322	0.94	0.94	0.18	0.29
361	0.86	0.85	0.23	0.31
522	0.88	0.87	0.21	0.3

In the table above each record represents an Arabic article used in our experiment. First article contains 134 unique words, proposed database covers the morphology of 94% article words, Translation of 96% of article words, synonyms of 30% of article words.

4. Conclusion

This paper presents a novel database for Arabic text mining applications. The database is called Al-Imam, which means the leader of all Arabic databases. The database contains 3588192 Diacritic words. These words generate about 17000000 Arabic English translation entries. The database contains 1644508 nouns and 1938817 verbs. Al-Imam database is linked with the English version of the WordNet database.

REFERENCES

[1] I. A. Al-Kharashi and M.W. Evans, "Comparing words, stems and roots as index terms in Arabic information retrieval System," *Journal of the American Society for Information Science (JASIS),* vol.5, no. 8, pp. 548-560, 1994.

[2] M. El-Kourdi, A. Bensaid and R. Tajje-eddine, "Automatic Arabic document categorization based on naïve Baye's Algorithm," in *Proc. of International Conference on Computational Linguistics, (COLING),* vol. 1, pp.51-58, Montreal, Canada, August, 2004.

[3] D. Evans, "Similarity based multilingual multi- document summarization," Columbia University, Technical Report CUCS--014-05, 2005.
[4] Euro WordNet Web Site: http://www.illc.uva.nl/Euro WordNet/ , (accessed 1 August 2009)
[5] Fellbaum (2004),WordNet 2.0, Available from: http://www.wordnet.princeton.edu/oldversions, (accessed 14 August 2009).
[6] D. Jurafsky and J. H. Martin, *Speech and Language Processing: an Introduction to Speech Recognition, Computational Linguistics and Natural Language Processing*, 2nd ed., Prentice Hall, 2008.
[7] Y. Kadri and J.Y. Nie," Effective Stemming for Arabic Information Retrieval " Laboratoire RALI, DIRO, Université de Montréal,2006.
[8] Y.S. Lee, " Morphological Analysis for Statistical Machine Translation," IBM T. J. Watson Research Center,
[9] R. Al-Shalabi, "Design and implementation of an Arabic morphological system to support natural language processing," Doctoral dissertation, Illinois Institute of Technology, Chicago, May 1996.
[10] F.S. Douzidia and G. Lapalme, "Lakhas, an Arabic summarization system," RALI-DIRO Université de Montréal, 2004. Yorktown Heights, NY 10598, 2004.
[11] F. Sadat, and N. Habash, "Arabic Preproc-essing Schemes for Statistical Machine Translation," Proceedings of Human Language Technology Conference of the NAACL.2006.

Recent Advances in Arabic Handwriting Recognition

Mostafa G. Mostafa*1, Mohamed F. Tolba**2

*Computer Science Department, Faculty of Computer & Information Sciences,

Ain Shams University, Abbassia 11566, Cairo, Egypt.
1mgmostafa@cis.asu.edu.eg

**Scientific Computing Department, Faculty of Computer & Information Sciences,

Ain Shams University, Abbassia 11566, Cairo, Egypt.
2mftolba@cis.asu.edu.eg

Abstract— **Arabic optical character recognition has been the subject of intensive research for the last four decades. The performance of Arabic machine-printed character recognition systems has developed considerably in the last decade. The recognition of Arabic handwriting, however, still an open research problem due to its substantial variation in appearance. This paper presents the recent advances in the last decade in both offline and online Arabic Handwriting recognition. Recent results for the online Arabic handwritten recognition show excellent recognition rate; A recognition rate of 100% was reported in the Last ICDAR recognition. However, despite there is a good recognition rate in case of constrained writing, the offline Arabic handwriting recognition still has many open problems that need more research. The main challenges, trends and contribution in the field of are discussed.**

1 INTRODUCTION

Document analysis and recognition (DAR) systems can contribute tremendously to the advancement of the automation process of the interaction between man and machine in many applications, including office automation, check verification, automatic mail-sorting, and a large variety of banking, business and data entry applications. Document analysis and recognition (DAR) systems decompose document content into two categories: Text and images. The text is then recognized automatically in order to be transformed from the pixel format into digital format for further processing. Optical character recognition is the core process for this transformation, and has been the subject of an active research in the past four decades [17], [46].

Nowadays, there exist successful DAR systems for the Arabic machine-printed documents [43]. This is due to the (relatively) standard features of the Arabic machine-printed text that facilitate the recognition process. To the contrary, Arabic handwritten text is quite different. Few Arabic DAR systems, with relatively acceptable recognition rate, are found for the analysis of the handwritten Arabic documents. This is an important research topic of today [25],[26], [27].

There are two recognition approaches that are distinguished according to the way handwriting data is collected: on-line and offline. In the online recognition, the data are captured during the writing process by a special pen on an electronic surface. In the latter, the data are acquired by a scanner after the writing process is over. In this case, the recognition of offline handwriting is more complex than the online case due to the presence of noise in the image acquisition process and the loss of temporal information such as the writing sequence and the velocity. This information is very helpful in a recognition process. Offline and on-line recognition systems are also discriminated by the applications they are devoted to. The offline recognition is dedicated to bank check processing, mail sorting, reading of commercial forms, etc, while the on-line recognition is mainly dedicated to pen computing industry and security domains such as signature verification and author authentication.

In this paper, we present the recent advances in the techniques of offline and online Arabic handwriting. Our survey is limited to the work done in the last decade, for previous works see [38], [9]. This paper is organized as follows. In Section 2, a quick background about the field, in which Arabic writing characteristics are discussed and an overview about the available databases are given. Section 3 presents the different methodologies introduced in the last decades to solve the online and the offline Arabic handwriting Recognition problem. We finally conclude the paper in Section 4.

2 BACKGROUND

In this section we summarize the main characteristics of the Arabic script and the major aspects of an optical character recognition system (OCR). An OCR system consists of four main stages: data acquisition, preprocessing, representation and features extraction, and recognition strategies. We also briefly overview most of the new trends of the techniques used in these stages. Public databases that are used in this field are also presented.

A. Characteristics of Arabic Script

Arabic script (machine-printed or handwritten) has special characteristics that differentiate it from many other live languages. The following points summarize the main characteristics of Arabic script (see Fig. 1 and Table 1 for an outline of the Arabic script and Arabic Alphabets):

- Arabic script is formed by an Alphabet that consists of 29 letters, and is written from right to left.
- Arabic text (machine printed or handwritten) is cursive and Arabic letters are normally connected on a baseline (a medium line in the Arabic word in which all the connections between the successive characters take place.)
- An Arabic letter might have up to four different shapes, depending on its position in the word. Table 1 shows the variation of shapes of the Arabic characters according to their positions in the word.
- Some Arabic characters may have exactly the same shape, and are distinguished from each other only by addition of diacritics. Namely, a dot, double dots, or triple dots. These dots may appear above or below the baseline. It is worth noting that any erosion or deletion of these dots results in a wrong classification of the character.
- Arabic words are formed by connecting some letters together. However, some Arabic characters are not connectable with the succeeding character. Therefore, if one or more of these characters exist in a word, the word is divided into two or more subwords. A subword can be one letter, two letters, or group of letters, as shown in Fig. 1.
- Another feature of the Arabic writing is the *ligature*. A ligature is the combination of two characters to form a unit shape, see Fig. 1. Ligatures occur only in some fonts, for example the two characters *Noon* ـنـ and *Meem* ـمـ can take the shape ـنمـ in the *"Simplified Arabic"* or the shape ﳒ in the *"Traditional Arabic"* as in Fig. 1. Ligatures form a challenge to most AOCR systems.
- Also two letters can *overlap* to raise a problem for the segmentation using simple horizontal projection.
- Furthermore, characters of the same font have different sizes, i.e. characters may have different widths even though the two characters have the same font and point size.
- *Fortunately*, Arabic characters can be represented using only 34 unique characters after removing the diacritics, and removing the Middle and End characters that have the same shape as the start characters, see Table 2.

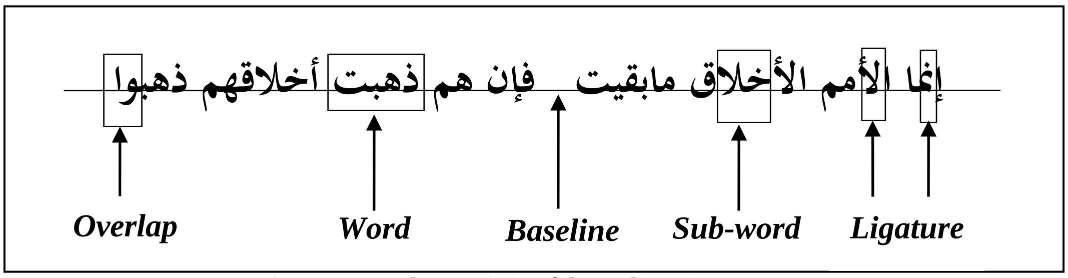


Figure 1. Some characteristics of the Arabic script.

B. Arabic Databases

Databases are cornerstone for any OCR system. They contain different samples that represents the class of objects the system intends to recognize. They are necessary resources to the advances of the research field, where they form a common ground for the performance evaluation of the different approaches proposed to solve the OCR problem. Four databases for the Arabic Optical Character Recognition (AOCR) are now available to the research communities. Namely: The IFN/ENIT database [35] The ARABASE database [8], and the ADAB database (). In the following we briefly describe such databases.

1) IFN/ENIT Database: Pechwitz *et. al.*) [35] introduced the IFN/ENIT database of Arabic handwritten names of Tunisian towns/villages. It contains more than 2200 binary images of handwriting sample forms from 411 writers, about 26,000 binary word images have been isolated from the forms and saved individually for ease of access. A ground truth file for each word in the database has been compiled. This file contains information about the word such as the position of the words base line, and information on the individual used characters in the word. The IFN/ENIT database can be downloaded from (http://www.ifnenit.com) **.**

2) *ARABASE Database:* Ben Amara *et. al.* [8] introduced the ARABASE database in 2005 for the research of Arabic offline and online handwriting and machine-printed text recognition. The database contains digital images of documents, text phrases, words/sub-words, isolated characters, digits, numerals, and signatures. The data corresponds to a variety of lexes (cities

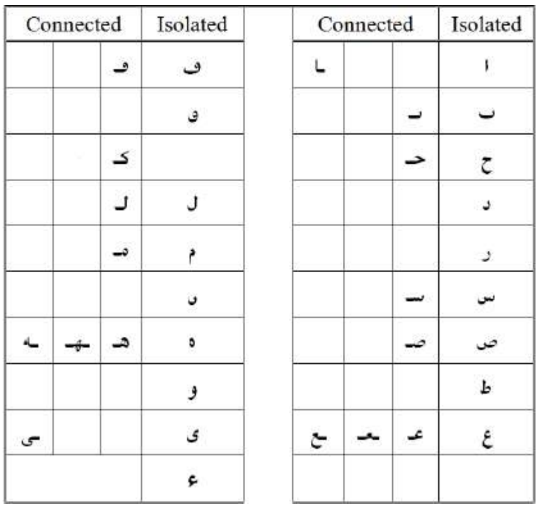
name, literal amounts, isolated characters, digits, free texts, etc.). Figure 2. Illustrate the different types of Arabic writing in the ARABASE database.

3) *ADAB databases:* The database ADAB (Arabic DAtaBase) [14],[15] was developed in a cooperation between the Institut für Nachrichtentechnik (IfN) and the Ecole Nationale d'Ingènieurs de Sfax (ENIS), Research Group on Intelligent Machines (REGIM), Sfax, Tunisia. The database in version 1.0 consists of 15,158 Arabic words handwritten by more than 130 different writers. The text written is from 937 Tunisian town/village names. The ADAB database is made available to the test the candidate systems submitted to the Arabic online handwriting recognition competition, ICDAR 2009.

TABLE 1. The different shapes of the Arabic characters that arise from the different position of the character within a word (E=End, M=Middle, B=Beginning, I=Isolated).

No.	Char. Name	E	M	B	I
1	*'Alif*	ـا	ـا	ا	ا
2	*Ba*	ـب	ـبـ	بـ	ب
3	*Ta*	ـت	ـتـ	تـ	ت
4	*Tha*	ـث	ـثـ	ثـ	ث
5	*Jeem*	ـج	ـجـ	جـ	ج
6	*H'a*	ـح	ـحـ	حـ	ح
7	*Kha*	ـخ	ـخـ	خـ	خ
8	*Dal*	ـد	ـد	د	د
9	*Thal*	ـذ	ـذ	ذ	ذ
10	*Ra*	ـر	ـر	ر	ر
11	*Zeen*	ـز	ـز	ز	ز
12	*Seen*	ـس	ـسـ	سـ	س
13	*Sheen*	ـش	ـشـ	شـ	ش
14	*S'ad*	ـص	ـصـ	صـ	ص
15	*Dhad*	ـض	ـضـ	ضـ	ض
16	*DTa*	ـط	ـطـ	طـ	ط
17	*DTha*	ـظ	ـظـ	ظـ	ظ
18	*Ein*	ـع	ـعـ	عـ	ع
19	*Ghein*	ـغ	ـغـ	غـ	غ
20	*Fa*	ـف	ـفـ	فـ	ف
21	*Qaf*	ـق	ـقـ	قـ	ق
22	*Kaf*	ـك	ـكـ	كـ	ك
23	*Lam*	ـل	ـلـ	لـ	ل
24	*Meem*	ـم	ـمـ	مـ	م
25	*Noon*	ـن	ـنـ	نـ	ن
26	*Ha*	ـه	ـهـ	هـ	ه
27	*Waw*	ـو	ـو	و	و
28	*Ya*	ـى	ـيـ	يـ	ى
29	*Hamza*	أ، إ، ؤ، ئ، ئـ			ء

TABLE 2: The 34 unique shapes of the Arabic characters.



Connected			Isolated	Connected			Isolated
		ڡـ	ڡ	ـا			ا
			ٯ			ٮـ	ٮ
		كـ				حـ	ح
		لـ	ل				د
		مـ	م				ر
			ں			سـ	س
ـه	ـهـ	هـ	ه			صـ	ص
			و				ط
ـى			ى	ـع	ـعـ	عـ	ع
			ء				

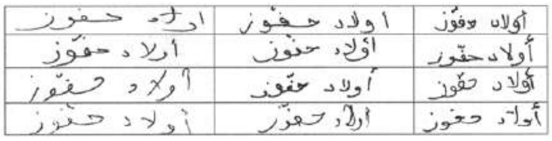


Figure 2. Illustration of a few entries in the IFN/ENIT database: (Pechwitz, 2002).

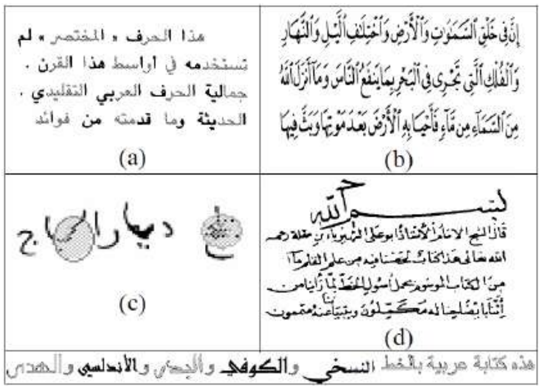


Figure 3. Illustration of a few characteristics of Arabic writing in the ARABASE database: : (a) Different sub-word of machine-printed text, (b) writing with diacritics, (c) free handwriting, (d) Historical handwritten documents, (e) multi-font machine-printed text, (Ben Amara, 2005).

C. CAHRACTER RECOGNITION SYSTEMS

The general model of Arabic optical character recognition (AOCR) systems can be described, likewise most of the pattern recognition systems, in terms of four stages: *preprocessing*, *segmentation*, *feature extraction*, and *classification*.

- In the *preprocessing* is a collection of operations that apply some digital filters that enhance the document image by reducing noise and distortion, and hence improve the recognition process. It may also include a *thinning* step to extract the *skeleton* of the word to facilitate the segmentation and the feature extraction stages.
- The *segmentation* stage decomposes the document into text and graphics, lines of a paragraph, and characters of a word. This stage is a cornerstone to most of the character-based method and the recognition rate depends solely on the results of this stage.
- The *feature extraction* stage analyzes a text segment and selects a set of features that can be used to uniquely identify the text segment.
- *Classification* is the main stage of any OCR system. It uses the extracted features from the previous stage to recognize the text segment according to different methods, which are discussed in the following sections.

Some systems use a further *post-processing* stage can be included to improves the recognition by refining words recognition by using context or word validation by using a dictionary search.

There are two methodologies which have been applied to both printed and handwritten Arabic character recognition, holistic and analytical methodologies. In the holistic methodology, there is no attempt to identify characters individually; the recognition is globally performed on the whole representation of words. Methods in this category include dynamic programming, neural networks, probabilistic framework, and expert systems. In the analytical strategies, words are considered a sequence of connected characters or small units which may be a part of a character. The recognition is then performed at the character or the small units level. In later segmentation step in order to segment a word into isolated characters. Having good segmentation technique is essential for having a more reliable Arabic DIA system.

Handwriting recognition systems can be classified, according to the data acquisition stage, into two main categories, namely, *online* and *offline* systems. Online recognition is performed as the text to be recognized is written. Therefore, the process of handwriting has to be captured online, e.g., using some pressure sensitive devices. They provide rich sequences of sensor data including geometrical positions of the stylus as well as temporal information about the writing process, which is the big advantage of online approaches. In contrast, offline recognition is performed after the text has been written. For this purpose, images of the handwriting are processed, which are captured using a scanner or a digital camera. In the following sections we summarize the reported result of both online and offline results that were published in the last decade.

3 ONLINE HANDWRITING RECOGNITION METHODS

It is commonly agreed that online handwriting recognition corresponds to the easier problem. Consequently, at least for certain application domains like pen-based input interfaces substantial progress has been achieved, se for example the promising results of the last Online Arabic Handwriting Recognition Competition [14],[15]. In fact, commercially available products suggest that the problem of online HWR can be considered as being close to solved [50]. In the three ICDAR competitions [25], [26], [27] specially the last one, interesting results have been reported for the online Arabic handwriting recognition. It was found that the best systems that show very high recognition rate are, on one hand, use neural networks based techniques, and on the other hand, the system does not use any normalization or feature extraction; they are the system works on the raw data directly [14],[15]. The proposed classification methods was mainly based on neural networks and Hidden Markov Models (HHM), the systems that are build using Neural Networks overcome systems that use Hidden Markov Models (HMM) [13] with less running time and more recognition accuracy. [17], [50]. In the following we summarize the results of the main contributions in online Arabic handwriting recognition.

Mezghani et al. developed a new representation used in online Arabic handwritten character recognition with Kohonen associative memory [28],[29],[30]. This representation based on statistics of features based on histograms of tangents and tangent differences at regularly spaced points along the character signal. For distance functions, they used Euclidean distance and investigated the Kullback-Leibler divergence and the Hellinger distance functions to measure the distance between the distributions. Furthermore, they performed some pruning and filtering operations on the trained memory to improve the classification accuracy. They used Kohonen neural network (SOM) in the recognition step that is trained using the proposed representation and compare its performance with K-nearest neighbor as benchmark.

Zafar *et al.* used backpropagation neural network (BPN) as a classifier in the second algorithm. They concentrated on sub-character primitive feature extraction technique that did not utilize resizing of characters. Direction encoding was used to shorten the characteristics of characters and they combined them to create a global feature vector. The database had 2000 characters collected from 40 subjects using tablet Summa Sketch III. Their script classifier requires training on different surfaces because the writing styles may vary considerably on these surfaces to increase the recognition rate (Zafar 2006).

In Mirvaziri *et. al.* [31], the authors investigated and evaluated several handwriting recognition algorithms with respect to the size of database, language and recognition rate Two of these algorithms deal with the online handwritten recognition. The first one uses a recurrent neural network (RNN) for its ability to process data with long time dependencies. They combined CTC network (Connectionist Temporal Classification) and LM (Language Model). CTC is an objective function designed for sequence labeling with RNNs but it does not require pre-segmented training data to transform the network outputs into labeling. They used IAM – On DB dataset that consists of pen trajectories collected from 221 different writers using a 'smart white board' as the training, validation and testing sets[18].

Razzak et al. [41] presents a segmentation-free approach for recognition of online Urdu handwritten script using hybrid classifier, HMM and fuzzy logic. Their trained data set consisting of HMMs for each stroke is further classified into 62 sub-patterns based on the primary stroke shape at the beginning and end using fuzzy rule. Fuzzy linguistic variables based on language structure are used to model features and provide suitable result for large variation in handwritten strokes. They applied data smoothing on chain code of the strokes to remove the zigzag path because the unsmooth data may produce problem for some features like cusp extracted from upward or downward sharp turning point as [42][43]. Twenty-six time variant structural and statistical features are extracted for the base strokes. Their experimental results show that the fuzzy classification into sub-patterns increases the efficiency and decreases the computational complexity due to reduction in data set size. The hybrid HMM–fuzzy technique is efficient for large and complex data set. It provided 87.6% and 74.1% for Nasta'liq and Nasakh, respectively, on 1800 ligatures.

In [22], the authors proposed two methods for recognizing handwritten Arabic and Chinese words and phrases. The first method, Symbolic Indirect Correlation (SIC), can be used in both online and offline recognition. The SIC method was introduced by the same authors in [35], [36]. It reassembles variable length segments of an unknown query that match similar segments of labeled reference words. Recognition is based on the correspondence between the order of the feature vectors and of the lexical transcript in both the query and the references. SIC implicitly incorporates language context in the form of letter n-grams. The second method, Style Constrained Classification (SCC), is based on the notion that the style (distortion or noise) of a character is a good predictor of the distortions arising in other characters, even of a different class, from the same source. It is adaptive in the sense that, with a long-enough field, its accuracy converges to that of a style-specific classifier trained on the writer of the unknown query

4 OFFLINE RECOGNITION METHODS

Offline recognition systems usually work on the scanned images of the documents. Therefore, the temporal information associated to the pin-point sequence, which are available to the online system, is now not available. The loss of such temporal information makes the offline OCR task harder than that of the online. Representation of the handwriting plays a crucial role in the performance of the recognition system. Accordingly, many representation techniques were proposed to extract a discriminative feature of the handwriting. These representations includes, structural [32], graph representation [33], etc. The classification methodologies proposed in the recent research are mainly Neural Networks [44], [48] and Hidden Markov field, many other methods were proposed, which we summarize the main work done in the last decade in this section.

Mahmoud and Mahmoud [24] used Hartley transform for the optical recognition of printed (not handwritten) Arabic characters. Their approach starts by extracting the contours of the primary parts of the characters. The Fast Hartley Transform (FHT) is then applied to the contours of each character and only 10 Hartley descriptors are extracted. The dots and holes of the character are also used besides the 10 descriptors to enhance the recognition rate. The authors reported a recognition rate of 97.3%, a rejection rate of 2% and an error rate of 0.7%. They also observed that their technique (based on Hartley descriptors) were comparable to the FFT-based Fourier descriptors in terms of recognition rate. The Hartley and FFT-based descriptors were found to give higher recognition rate than the Modified Fourier Spectrum (MFS) descriptors. In all their experiments, they used the nearest neighbor classifier.

Ziaratban and Faez [53] presented an approach for deslanting Farsi/Arabic scripts. In the first phase, this approach estimate the overall tilt of a handwritten word based on directional filters. After overall deslanting, they apply non-uniform slant estimation algorithm to compute the remaining slant of each near-vertical stroke of the word separately. A non-uniform slant correction algorithm is then used to reduce the remaining slants of each candidate stroke keeping the distortions of other strokes of the word at a minimum level. When the authors compared this approach and other prevalent methods (based on chain codes and vertical projection), they found that the proposed gives the least estimation error and runs the fastest.

Plotz and Fink [39] surveyed the techniques of offline handwriting recognition that are based on Markov models (MMs). They started by describing the architecture typical to MM-based offline handwriting recognition systems. Then, they presented a review of the solutions proposed in the literature for the open problems associated with the use of MMs for handwriting recognition.

Benjelil et al. [11] proposed a system for page segmentation and classification of complex documents. The system is to be used to determine the layout of the pages of the input document. This information is fed into the OCR system along with the input document. The proposed system is based on steerable pyramid transform. The features extracted from pyramid sub-bands are used to locate and classify regions into text (either machine-printed or handwritten) and non-text (images, graphics, drawings or paintings) that may exist in some noise-infected, deformed, multilingual, multi-script document images. Such documents can contain tabular structures, logos, stamps, handwritten script blocks, photographs, etc. They used a dataset of 1,000 official complex document images to test their system against other state-of-the-art methods. The results showed that their system achieved higher accuracy than its competitors.

Al-Shaher and Hancock [3] presented a statistical approach for recognizing 2D shapes which are represented as an arrangement of curves or strokes. The approach is a hierarchical one which mixes geometric and symbolic information in a three-layer architecture. Each curve primitive is represented using a point-distribution model which describes how its shape varies over a set of training data. Shapes are decomposed into an arrangement of primitives and the global shape representation has two components. The first of these is a second point distribution model that is used to represent the geometric arrangement of the curve centre-points. The second component is a string of stroke labels that represents the symbolic arrangement of strokes. They use recover the stroke parameters, shape-alignment parameters and stroke labels by applying the expectation maximization (EM) algorithm. The authors applied the resulting shape-recognition method to handwritten Arabic character recognition and achieved a recognition accuracy of 97%.

Lopresti et al. [22] proposed two methods, Symbolic Indirect Correlation (SIC) and Style Constrained Classification (SCC), for recognizing handwritten Arabic and Chinese words and phrases. SIC reassembles variable-length segments of an unknown query that match similar segments of labeled reference words. For recognition, it uses the correspondence between the order of the feature vectors and of the lexical transcript in both the query and the references. SIC implicitly incorporates language context in the form of letter n-grams. The other method, SCC, is based on the notion that the style (distortion or noise) of a character is a good predictor of the distortions arising in other characters, even of a different class, from the same source. It is adaptive in the sense that, with a long-enough field, its accuracy converges to that of a style-specific classifier trained on the writer of the unknown query. Neither SIC nor SCC requires the query words to appear among the references.

Al-zoubaidy [5] described a methodology for feature selection in unsupervised learning for application in Arabic handwritten character recognition. The methodology uses a multiobjective genetic algorithm that minimizes the number of features. A validity index that measures the quality of clusters have been used to guide the search towards the more discriminate features and the best number of clusters. Experimental analysis of the method on Arabic handwritten character recognition indicated an improvement in the recognition speed as well as the recognition accuracy. It achieved a 30-50% reduction of features with trivial drop in recognition accuracy. The obtained identification accuracy was 95% for the training set, 89% for the validation set, and 85.3% for the test set.

Amrouch *et al.* [7] addressed the offline recognition of isolated Arabic handwritten characters. Their system relied on Hidden Markov Models (HMMs). The Hough accumulator of the character image in this system is partitioned into equal horizontal bands to be used to extract directional information. This information is translated into sequences of observations that are used to train the model for each character during the learning step. The system was experimented on 42 Arabic handwritten isolated characters extracted from the base of characters. 50% of these served for the phase of learning, and 50% for the tests. The proposed system achieved a recognition rate of 85.71%.

Amin in [6] applied Inductive Logic Programming (ILP) to the automatic recognition of Arabic characters. His system used a structural approach for feature extraction (based on structure primitives such as curves, straight lines and loops in similar manner to which human begins describe characters geometrically). The proposed system was tested on a sample of handwritten characters from several individuals whose writing ranged from acceptable to poor in quality and the average correct recognitions rate obtained using cross-validation was 86.65% while the rejection rate was 96.95%

Alshebeili et al.[4] presented an Arabic character recognition algorithm using 1-D slices of the character Fourier spectrum. It estimates the Fourier spectrum of the character's projections on the X- and Y-axes, and the features are extracted from this 2-D spectrum. The features of 10 sets of characters were used as model features. The features obtained this way are invariant to changes in scale, orientation and shift. To classify an input character, the algorithm extracts its feature vector and measures its distance from the model features. The algorithm outputs the model whose feature vector is closest to the feature vector of the input character. Experimental results have shown that the presented algorithm is capable of recognizing Arabic characters with a recognition rate of 99.06%, using 10 features of the X-projection. Taking 10 additional features from the Y-projection improves the recogntion rate to 99.94%.

Mozaffari et al. [34] proposed a fast method for extracting dots from cursive Farsi/Arabic handwriting and suggested a technique for utilizing such dots in lexicon reduction. The technique involves extraction and representation of number and position of dots from offline handwritten words to eliminate unlikely candidates. The system was tested on using a set of 12,000 handwritten word images and a lexicon reduction of 93% with accuracy of 85% was obtained. The incorporation of the proposed lexicon reduction algorithm achieved a speedup factor of 2 as well as 13% improvement in recognition rate.

Farooq et al. [16] proposed a phrase-based post-processing technique for correcting the output of an arbirary OCR. The system treats the OCR as a black box and adapts statistical machine translation techniques to correct the output of the OCR. The system was able to improve the recognition rate of an Arabic OCR developed by Sakhr from 71.1% to 84.8% on a dataset containing around 400 scanned documents.

Khorsheed [21] proposed a method that does not require segmentation into characters, and is applied to cursive Arabic script. The method trains a single hidden Markov model (HMM) with the structural features extracted from the manuscript words. These structural features are obtained by decomposing the skeleton graph of the word into a sequence of links (in the order in which the word is written). Then, each link is further broken into small line segments using line approximation. The line segment sequence is transferred into a sequence of discrete symbols using vector quantization and finally presented to the HMM to obtain an sequence of the letters associated with the input pattern. To assess the performance of the proposed method, samples extracted from a historical handwritten manuscript were used (written by a single person). The system achieved a recognition rate of 81%. The use of an Arabic spell-checker helped increase the recognition rate to 97%. However, these recognition rates are expected to decrease when including more handwritten fonts.

Abdulla et al.[1] described a segmentation algorithm using Rotational Invariant Segments Features (RISF). The algorithm uses a dynamic feature extraction technique to evaluate a large set of curved segments or strokes through the image of the input Arabic word or subword. It then nominates a small “optimal” subset of cuts for segmentation. All the directions of stroke are converted to two main segments: '+' and w'-' RISF. The RISF algorithm was tested on databases designed by the authors and achieved a segmentation rates between 90.58% and 95.66%.

Xiu et al. [51] presented a probabilistic segmentation model. It starts by conducting a contour-based over-segmentation to cut the word image into graphemes. The graphemes are sorted by the algorithm into 3 queues, which are character main parts, sub-parts (diacritics) above and below main parts, respectively. The confidence for each character is calculated by the probabilistic model, taking into account both of the recognizer output and the geometric confidence besides with logical constraint. Then, the global optimization is conducted to find optimal cutting path, taking weighted average of character confidences as objective function. The method was tested on various writing styles and an average recognition rate of 59.2% was achieved.

Kessentini et al. [20] presented a multi-stream feature approach for offline handwritten word recognition. The proposed approach combines low-level feature streams, namely (a) density-based features extracted from 2 different sliding windows with different widths, and (b) contour-based features extracted from upper and lower contours. The approach was tested on a Latin script database as well as an Arabic script one. The recognition performance was 89.8% for the Latin database and 79.8% for the Arabic one.

Al-Ohali and Brook [2] presented a holistic technique for classifying and retrieving historical Arabic handwritten (HAH) documents. The algorithm proceeds as follows. First, the HAH manuscript’s image is segmented into words and each word is segmented into its connected parts. To improve the segmentation, the overlap between the adjacent connected parts of a single word is reduced by a stretching algorithm that increases the gap between the connected parts. After that, a variety of structural and statistical features, devised specifically for Arabic text, are extracted from the connected parts and then combined into one consolidated feature vector representing the word as a whole. A neural network is then used to learn and classify the input vectors into word classes. These classes are then utilized to retrieve HAH manuscripts.
Benouareth et al. [12] described an offline unconstrained handwritten Arabic word recognition system based on segmentation-free approach and semi-continuous hidden Markov models (SCHMMs) with explicit state duration. The described system performs word recognition by using a proposed sliding window approach based on vertical projection histogram analysis of the

word and extracting a pertinent set of statistical and structural features from the word image. The authors compared three distributions (Gamma, Gauss and Poisson) for the explicit state duration modeling where the Gamma distribution achieved the best recognition accuracy. Several experiments have been performed in which the proposed system achieved recognition rates ranging from 90.2 to 97.50.

5 CONCLUSION

In this paper, we presented the main advances in the online and offline Arabic handwriting recognition, and some of the databases available to the research communities for this task. The online Arabic handwriting recognition systems have recently achieved a very good advancement. Most of the proposed classification methods were mainly based on neural networks and Hidden Markov Models (HHM), the systems that are build using Neural Networks overcome systems that use Hidden Markov Models (HMM). Interestingly, some systems (Vision Objects, the winning system in the Last ICDAR 2009 competition) reported a recognition rate of 100% with fast processing speed. It worth noting that the system that achieved such excellent performance does not use any normalization or feature extraction; the system works on the raw data directly. These results indicates that the online handwriting recognition is almost a solved problem.

Offline recognition systems usually solve harder problem than the online problem. This is due to the lack of the temporal information associated to the pin-point sequence, which are available to the online system. Representation of the handwriting plays a crucial role in the performance of the offline recognition system. Accordingly, many representation techniques were proposed to uniquely represent the different parts of the handwriting text. These representations include structural graph representation. The classification methodologies proposed in the recent research are mainly Neural Networks, Hidden Markov field; many other methods were also proposed. Most of the proposed systems use segmentation-based methodologies. Segmentation is the main source for erroneous results; this is due to the cursive nature of the Arabic script and the different styles of the writers. For that, the best performance achieved until now for the offline systems id about 97% for constraint writing, and fell into about 60% in case of free-writing.

REFERENCES

[1] Abdulla, A. Al-Nassiri and R. Abdul Salam, "Offline Arabic Handwritten word segmentation using rotational invariant Segments Features", *The International Arab Journal of Information Technology*, vol. 5. no. 2, pp. 200, 2008.

[2] Z. Al Aghbari, and S. Brook, "HAH manuscripts: A holistic paradigm for classifying and retrieving historical Arabic handwritten documents", *Expert Systems with Applications*, Vol. 36, No. 8, pp.10942-10951, 2009.

[3] Al-Shaher and E.R. Hancock, "Arabic Character Recognition Using Structural Shape Decomposition", Lecture Notes in Computer Science, Computer Analysis of Images and Patterns, Vol. 2756, pp. 478-486, 2003.

[4] S.A. Alshebeili, A.F. Nabawi and S.A. Mahmoud, "Arabic character recognition using 1-D slices of the character spectrum", Signal Processing, vol. 56, No. 1, pp. 59-75, 1997.

[5] L.M. Al-zoubaidy, "Efficient Genetic Algorithms for Arabic Handwritten Characters Recognition", Advances in Soft Computing, Applications of Soft Computing, Vol. 36, pp. 3-14, 2006.

[6] Amin, "Recognition of Hand-Printed Characters Based on Structural Description and Inductive Logic Programming", Pattern Recognition Letters, Vol. 24, pp. 3187-3196, 2003.

[7] M. Amrouch, M. Elyassa, A. Rachidi and D. Mammass, "Off-Line Arabic Handwritten Characters Recognition Based on a Hidden Markov Models", Lecture Notes in Computer Science, Image and Signal Processing, Vol. 5099, pp. 447-454, 2008.

[8] N. Ben Amara, O. Mazhoud, N. Bouzrara and N., Ellouze, "ARABASE: A Relational Database for Arabic OCR Systems", The International Arab Journal of Information Technology, vol. 2, No. 4, p. 259, 2005.

[9] N. Arica and F.T. Yarman-Vural, "An overview of character recognition focused on off-line handwriting". IEEE Trans. Syst. Man Cybern. C Appl. Vol 31, no. 2, pp. 216–232, 2001.

[10] S. M. Awaidah, S. A. Mahmoud. A multiple feature/resolution scheme to Arabic (Indian) numerals recognition using hidden Markov models. Signal Processing, 89, 1176–1184, (2009).

[11] M. Benjelil, S. Kanoun, R. Mullot and A.M. Alimi, "Complex documents images segmentation based on steerable pyramid features", International Journal on Document Analysis and Recognition, Vol. 13, No. 3, pp. 209-228, 2010.

[12] Benouareth, A. Ennaji and M. Sellami, M., "Semi-continuous HMMs with explicit state duration for unconstrained Arabic word modeling and recognition". Pattern Recognition Letters, vol. 29, No. 12, pp. 1742-1752, 2008.

[13] F. Biadsy, J. El-Sana and N. Habash, "Online arabic handwriting recognition using hidden markov models", in Proc. of the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR), pp. 85–90, 2006.

[14] H. El Abed and V. Märgner, "ICDAR 2009-Arabic handwriting recognition competition", International Journal on Document Analysis and Recognition, pp. 1433–2833, 2010.

[15] H. El Abed, M. Kherallah, V. Märgner and A.M. Alimi, "On-line Arabic handwriting recognition competition: ADAB database and participating systems". International Journal on Document Analysis and Recognition, pp. 1388-1392, 2010.

[16] F. Farooq, D. Jose and V. Govindaraju, "Phrase-based correction model for improving handwriting recognition accuracies". Pattern Recognition, vol. 42, No. 12, pp. 3271-3277, 2009.

[17] H. Fujisawa, "Forty years of research in character and document recognition—an industrial perspective", Pattern Recognition. Vol 41, pp. 2435–2446, 2008.

[18] Graves, S. Fernàndez and J. Schmidhuber, "Multi-dimensional recurrent neural networks", in Proc. of the International Conference on Artificial Neural Networks, Porto, Portugal, 2007.

[19] Graves, S. Fernandez, M. Liwicki, H. Bunke and J. Schmidhuber, "Unconstrained online handwriting recognition with recurrent neural networks". Advances in Neural Information Processing Systems 21, NIPS'21, pp 577–584, 2008.

[20] Y. Kessentini, P. Thierry and A. Ben Hamadou, "Off-line handwritten word recognition using multi-stream hidden Markov models", Pattern Recognition Letters, vol. 31, No. 1, pp. 60-70, 2010.

[21] M.S. Khorsheed, "Recognising Handwritten Arabic Manuscripts Using a Single Hidden Markov Model", Pattern Recognition Letters, vol. 24, pp. 2235-2242, 2003

[22] D. Lopresti, G. Nagy, S. Seth and X. Zhang, "Multi-character Field Recognition for Arabic and Chinese Handwriting", Lecture Notes in Computer Science, Vol. 4768, Arabic and Chinese Handwriting Recognition, pp 218–230, 2008.

[23] L.M. Lorigo, and V. Govindaraju, "Offline Arabic Handwriting Recognition: A survey", IEEE Trans. on Pat. Anal. and Mach. Int (PAMI), Vol 28< no. 5, pp. 712–724, 2006.

[24] S.A. Mahmoud, and A.S. Mahmoud, "The use of Hartley transform in OCR with application to printed Arabic character recognition". Pattern Analysis & Applications, vol. 12, No. 4, pp. 353-365, 2009.

[25] V. Margner, M. Pechwitz, and H. El Abed, "ICDAR 2005—Arabic handwriting recognition competition", in Proc. of 8th International Conference on Document Analysis and Recognition (ICDAR), vol. 1, pp. 70–74, 2005.

[26] V. Margner and H. El Abed, "ICDAR 2007—Arabic handwriting recognition competition", in Proc. of the 9th International Conference on Document Analysis and Recognition (ICDAR), vol. 2, pp. 1274–1278, 2007.

[27] V. Margner and H. El Abed, "ICDAR 2009 Arabic handwriting recognition competition", in Proc. of the 10th International Conference on Document Analysis and Recognition (ICDAR), vol. 3, pp. 1383–1387, 2009.

[28] N. Mezghani, A. Mitiche, and M. Cheriet, "A new representation of shape and its use for high performance in online Arabic character recognition by an associative memory", International Journal on Document Analysis and Recognition (IJDAR), Vol 7, no 4, pp. 201-210, 2005.

[29] N. Mezghani, A. Mitiche and M. Cheriet, "A new representation of shape and its use for high performance in online Arabic character recognition by an associative memory", International Journal of Document Analysis, Vol. 7, no 4, pp 201–210, 2005.

[30] N. Mezghani and A. Mitiche, "A Gibbsian Kohonen Network for Online Arabic Character Recognition". Lecture Notes in Computer Science, Advances in Visual Computing, Vol. 5359, pp. 493–500, 2008.

[31] H. Mirvaziri, M.M. Javidi and N. Mansouri, "Handwriting Recognition Algorithm in Different Languages: Survey", Lecture Notes in Computer Science, Vol. 5857, Visual Informatics: Bridging Research and Practice, pp. 487–497, 2009.

[32] M. G. Mostafa, "An Adaptive Algorithm for the Automatic Segmentation of Arabic Characters", in Proc. of the 17th National Conference on Computers, Al-Madinah Al-Munawwarah, Saudi Arabia, December 15-18, 2003.

[33] M.G. Mostafa, "Optical Character Recognition of Handwritten Arabic Numerals Using Structured Graph", in Proc. of the 3rd International Conference on Intelligent Computing and Information Systems, Cairo, Egypt 15-18 March, pp. 275-280, 2007.

[34] S. Mozaffari, K. Faez, V. Margner, and H. El-Abed, "Lexicon reduction using dots for off-line Farsi/Arabic handwritten word recognition", Pattern Recognition Letters, vol. 29, No. 6, pp. 724-734, 2008.

[35] G. Nagy, S.C. Seth, S.K. Mehta and Y. Lin, "Indirect Symbolic Correlation Approach to Unsegmented Text Recognition", in Proc. of Conference on Computer Vision and Pattern Recognition, Workshop on Document Image Analysis and Retrieval (DIAR 2003), Madison, WI, pp. 22–32, 2003.

[36] G. Nagy, D. Lopresti, M. Krishnamoorthy, Y. Lin, S. Seth and S. Mehta, "A Nonparametric classifier for unsegmented text", in Proc. of IS&T-SPIE International Symposium on Document Recognition and Retrieval, San Jose, pp. 102–108, 2004.

[37] M. Pechwitz, S.S. Maddouri, V. Märgner, N. Ellouze, and H. Amiri, "IFN/ENIT-DATABASE OF HANDWRITTEN ARABIC WORDS" , in the 7th Colloque International Francophone sur l'Ecrit et le Document , CIFED 2002, Hammamet, Tunis, Oct. 21-23, 2002.

[38] R. Plamondon, and S.N. Srihari, "On-line and off-line handwriting recognition: a comprehensive survey", IEEE Trans. Pattern Anal. Mach. Intell., Vol 22, no. 1, pp. 63–84, 2000.

[39] T. Plotz and J.A. Fink. "Markov models for offline handwriting recognition: a survey", International Journal on Document Analysis and Recognition, Vol. 12, No. 4, pp. 269-298, 2009.

[40] M.I. Razzak, S.A. Hussain and M. Sher, "Combining online and offline preprocessing for online Urdu character recognition", in Proc. of International Multiconference of Engineers and Computer Scientists 2009 (IMECS 09), Hong Kong, 2009.

[41] M.I. Razzak, S.A. Hussain and M. Sher, "Numeral recognition for Urdu script in unconstrained environment", in Proc. of International Conference on Engineering and Technology, Islamabad, 2009.

[42] M.I. Razzak, F. Anwar, S.A. Husain, A. Belaid and M. Sher, "HMM and fuzzy logic: A hybrid approach for online Urdu script-based languages' character recognition", Knowledge-Based Systems, Vol. 23, no. 8, pp. 914–923, 2010.

[43] Sakhr® OCR System. Web Site: http://www.sakhr.com/OCR.aspx (Accessed November 2010).

[44] M. F. Tolba and G. M. Abdul Moty, "A Comprehensive Survey on Arabic Optical Characters Recognition," The 1st International Conference of Linguistic Engineering, Cairo, Egypt, October 1998.

[45] M. F. Tolba and S. K. Fathy. "A Structural Approach of an Arabic Character Recognition," The 2nd International Conference of Linguistic Engineering, Cairo, Egypt, October 1998.

[46] M. F. Tolba, G. M. Abdul Moty and A. M. Mahmoud, “Self-Organizing feature Maps For Arabic Sub-Words Recognition,” The International Conference on Industrial Electronics, IETA2001, Electronics Research Institute (ETI), IEEE Industrial Electronics Society & University of Bridgeport, Cairo, Egypt, December 2001.

[47] M.F. Tolba, G.M. Abdul Moty and A.M. Mahmoud, "Segmentation Free Approach for Printed Arabic Text Recognition", International Journal of Computers and Their Applications (ISCA), Vol 10, no. 2, pp. 94-102, 2003.

[48] M. F. Tolba, M. S. Abdel Wahab, I. A. Taha and A. M. Al-Shishtawy, “A Secure Grid Enabled Signature Verification System,” 2nd International Conference on Intelligent Computing and Information Systems (IJICIS), Cairo, Egypt, March 2005.

[49] M.F. Tolba, "Arabic Optical Character Recognition: State of the Art", The Nineth International Conference on Language Engineering, Cairo, Egypt, 2009.

[50] Vision Objects. Myscript handwriting recognition engine. Web Site: http://www.visionobjects.com/handwriting-recognition/how-doesmyscript-work/ (Accessed November 2009).

[51] P. Xiu, L. Peng, X. Ding and H. Wang, "Offline Handwritten Arabic Character Segmentation with Probabilistic Model", Lecture Notes in Computer Science, vol. 3872, Document Analysis Systems VII, pp. 402-412, 2006.

[52] M.F. Zafar, D. Mohamad and M.M. Anwar, "Recognition of online isolated handwritten characters by backpropagation neural nets using sub-character primitive features", in Proc. of 10th IEEE International Conference on Information Technology, INMIC 2006, Islamabad, Pakistan, pp. 157–162, 2006.

[53] M. Ziaratban and K. Faez, "Non-uniform slant estimation and correction for Farsi/Arabic handwritten words", International Journal on Document Analysis and Recognition (IJDAR), Vol. 12, no. 4, pp. 49–267, 2009.

An NLP-Based Fully Distributed Arabic Search Engine (1)

Taghride Anbar

Ain Shams University, Al-Alson Faculty

Taghride@coltec.net

Abstract

This prototype search engine is sponsored by ITIDA (Information Technology Industry Development Agency). It is one of ITAC projects (Information Technology Collaboration), of the type ARP program (Advanced Research Project). ITAC projects aim at "promoting Industry/Universities collaboration". The partners in this project are "Faculty of Computer & Information Sciences" at Ain-Shams University, and COLTEC (Computer & Language Technology) Company. Coltec's role is to build a single-CPU NLP-based Arabic search engine. The role of the Faculty of Computer & Information Sciences is to set up the system as a web search engine; this includes developing parallel web crawler and page ranking.

Our paper aims at representing the NLP techniques introduced to this prototype search engine; accordingly, it is out of the scope of the paper to characterize efforts and techniques applied on the different components of the system.

The target of our NLP techniques is to reduce as much as possible the linguistic distortion that affects both indexing and searching documents. Before introducing our NLP work, we discussed the methodology of automatic suffix splitting and illustrated how far it is inconvenient for processing Arabic. In the prototype search engine, the applied NLP techniques handle two sources of linguistic distortion; namely the writing and the affixation issues.

I. Introduction

Search engines are one of the main topics related to the domain of information retrieval, sometimes, named text retrieval. The continuous stream of textual contents on the web creates urgent need to search engines: no user can browse all of the web documents to get information on a certain concept. The user has to depend on a system that is capable of performing this task automatically, quickly and accurately. The user has to represent the required concept in a suitable word, a group of words, or a phrase. Doing that, a query has been built. A list of search results points to the documents that include the query word(s); each item in the list points to a document. When clicking on any item, the connected document opens. In few words, a web search engine aims at searching in a huge pool of natural language documents to return the set of documents whose content matches the query subject. Normally, the documents retrieved in a list are arranged in a way that the documents evaluated by the system as most like the query come on the top of the list. The statistical method is the most common method applied in Search engines.

II. Indexing & Querying

Though the diversity of methodology, design and structure of search engines, two components are basic in any search system: the indexing and the querying components.

The indexing component is an essential subsystem of the Web Search Engine. From documents repository, it extracts both content and attributes information and merges them into indexes, which facilitates fast and accurate searches. The Indexing component should be capable of reading and extracting information from various formats of files. In few words, Indexing is the overall process of extracting information, creating index entries, and merging them into a large index that contains the sorted terms and their locations within the pool

of indexed documents. The index unit is designed to execute all its tasks automatically and continuously so that new updates being included and replaced old documents being omitted.
Without an indexing system, the application would scan every document in the document repository, which requires considerable time and computing power.

Among other preparatory operations that the indexing process executes, there are three operations in direct relation to natural language;

1- Tokenization that returns texts into lists of words
2- Exclusion of stop words (relational or empty words) from being indexed: Prepositions, articles, pronouns… are examples of empty words. Such words - contrary to the full words that carry lexical meanings - have no lexical content. They are functional words that indicate relations among full words. Stop words are of high frequency; they represent about one third of text content. If one third of the words in a document are excluded from the indexing process, the indexing time and size will surely improve, meanwhile the content will not be affected. In addition, no user builds a query about a stop word. Based on these facts, almost all search engines do not include stop words in the index. The exclusion process is performed by the help of a pre-defined list of stop words.
3- Returning the different forms of a word to their basic lexical item (an operation known as stemming): words are the main way to express concepts, thus, when a user formulates a query, he means inquiring a concept. The search, in return, uses the query word(s) as a means to search the concept. In natural languages, full words are flexible; they may change their forms without changing their conceptual content. In English, such changes mostly take place by adding suffixes: suffixes change the lexical item (stem) “play” to: plays, played, playing… If the search is performed on words, without the stemming operation, only, the form of the query word will be included in the search results, the other forms will be dropped though they express the same concept. The word suffixes are automatically removed according to a pre-defined list. Indexing stems guarantees -to a large extent- that the required concept is widely covered independently from the linguistic variations. This methodology of automatic Affix removal suits perfectly the English languages; the reason is that the morphological system of English is relatively simple and limited. Meanwhile, This method does not suit other European languages; the German language that widely uses compound nouns is an example.

The querying component provides the search functions related to the index created by the Indexing component. The output of this component represents a list of hits resulted by an inserted query with chosen options. This list of search results are ordered by the Page Ranking module and viewed on the final interface. To generate the text snippets of the results, the caller subsystem should retrieve the matched documents from the document repository and extract the snippets according to the hit list.

The effectiveness of different search engines submits to continuous evaluation. The common evaluation method is mapping **precision** against **recall**. TREC (Text Retrieval Conference) is one of the most famous and important workshops in this domain.

III. NLP Track at TREC-5

The Text REtrieval Conference (TREC), is co-sponsored by the National Institute of Standards and Technology (NIST), U.S. Department of Defense, Information Technology Laboratory's (ITL), Intelligence Advanced Research Projects Activity (IARPA), and Retrieval Group of the Information Access Division (IAD).
The first conference took place in November 1992. “its purpose was to support research within the

information retrieval community by providing the infrastructure necessary for large-scale evaluation of text retrieval methodologies…TREC has also sponsored the first large-scale evaluations of the retrieval of non-English (Spanish and Chinese) documents"(TREC homepage: overview).
In 1996, one of the tracks was dedicated to examine the impact of NLP techniques on text retrieval. Tomek Strzalkowski summarized this track from his notes and Karen Spark Jones notes; "NLP track has been organized for the first time at TREC-5 to provide a more focused look at how NLP techniques can help in achieving better performance in information retrieval…Five teams participated in this NLP track" (http://www.dtic.mil).
The conclusions are "This NLP track demonstrated that natural language processing techniques have solid but limited impact on the quality of text retrieval, particularly precision. Techniques aimed at producing higher quality queries, e.g., query expansion, constraints, appear to be more effective than those aimed primarily at obtaining improved indexing of database documents. More work is needed before more substantial gains can be seen" (http://www.dtic.mil).

We judge these conclusions as not surprising; they are predictable. The language, subject to the NLP work was English; a language with relatively simple linguistic structure. We believe that the processing operations (mentioned above) are suitable and sufficient for search purposes; additional interference will not be of remarkable advantage, it might degrade the search effectiveness.
Are the conclusions of this track related specifically to English or extends to other languages even those with complicated structures? The answer of this question was not apparent in Strzalkowski's report. In all cases, Since 1996, no other track was dedicated in TREC events to the NLP.

IV. <u>Linguistic distortion in Arabic</u>

The affixional system in Arabic is so rich that the variations of one lexical item like "مناضل "extends to about 2000 words: " المناضل - ومناضل - للمناضلين - مناضلوهم...". Such affixes do not change the word's lexical content; they add to it detailed functional meanings. For human usage such richness is really fruitful; it allows the speaker\writer to add as many details as required to the lexical content. For most of the automatic applications, such details generate what we call "linguistic distortion"; search engines target searching concepts through words; If the language affixation system produces loads of different words expressing one and the same concept, the accuracy of search process will be highly affected. Linguistic distortion elimination by applying the automatic suffix splitting methodology fails when dealing with Arabic:

1- The affixation system, in Arabic, is complicated. It is not just single suffixes:
   up to three prefixes can combine together before a lexical item: "و بـ الـ بيت",
   two suffixes may follow a lexical item: " مدرسـ و هم ",
   lexical items accept compound prefixes and suffixes "فـ لـ حديقـ تـ يـ كم "
2- Superficial segmentation misses the prefix\suffix validity which may lead to wrong segmentation: لـعبدون - تـبعيـة …
3- Some characters look like affixes, while they are part of the lexical item: نـبيـه فـريد أوان …
4- Affix\ lexical items validation has to be pre-defined. It is not an automatic act:
   the lexical item : مهندس accepts the suffixes "ون ، ة", while " ولد " does not accept those suffixes.

In addition to mentioned above, there are other types of linguistic noise (specific to Arabic) that affects indexing & search processes:

a- In different languages, defining stop words is a simple task: "the, from, in, when" can easily be judged as stop words. In Arabic, there are full words whose orthography is identical with that of stop words: (stop word) منْ <> من (verb), (stop word) عنْ <> عن (verb), (stop word) علىَ <> على (proper noun) …
b- The Interaction between Affixes and lexical items may cause a kind of ambiguity:
   التهم → الـتهم & اِلْتَهَمَ, فواصل → فـواصَلَ & فواصِلَ بعيد → بـعيـد & بَعيـد

c- The absence of diacritics “تشكيل” causes another type of ambiguity:
   مستقبِل <> مستقبَل , مُنازل <> مَنازل , شَعْر <> شِعر

d- The lexical item changes orthographically in certain cases: موعد \ مواعيد - وظيفة \ وظائف

e- The Arabic orthographic system is full of writing details which result in: first; one single lexical item may have several writing variations: **سماء \ سمائـ \ سماؤ,** second; most of non-professionals in Arabic linguistics commit many errors when writing texts. The on the web Arabic e-contents confirm this claim.

The issues that cause linguistic distortion in relation to Arabic search engines can be classified in four categories: affixation, inflection, writing, and ambiguity.

V. <u>Arabic NLP Engine</u>

Solving these issues necessitate developing a powerful NLP engine that adequately addresses these problems. The NLP engine implemented in the prototype search engine overcomes properly the problems related to affixation and inflection. The writing issues are - partially but not totally- resolved. Dealing with linguistic ambiguities is the most complicated problem that NLP engines encounter in all languages; a first step solution has been introduced to the system.
The NLP engine introduced to this system is composed of three main units:

- The system's lexicon whose backbone is the lexical items. Each item is supplied with the linguistic data required for this application. Part of the data points to the affixes that the lexical item accepts.
- The Word Identifier tool whose function is to analyze each token, to extract the lexical item and the connected affixes, and to check the correctness of the token's components. This tool is so powerful that it verifies a word in 1/800 millisecond. In other words, it verifies 800 000 words/second.
- A set of simple rules that help with organizing the NLP processing and applying the required functions.

Affixional Options

Searching Arabic queries in most of the current web search engines begins by searching the query as being written, sometimes and not always, the word initial character(s) resembling one of the listed prefixes, is\are removed; the remaining part is included in the search process without inclusion of other acceptable affixes:
Two options are applied in regard to the affixation:.

1- **Exact Search option** where the query is searched as being written. This technique does not include affixes in the search process. It suits searching proper nouns in European languages because the equivalents of Arabic prefixes are written in these languages as independent words: “the, and, then…’; thus the search resulted will not be distorted by word variations. When applying the exact search, in Arabic, on a query like “**طه حسين**”, the search results will miss structures like “ **وطه حسين , لطه حسين , كطه حسين** “. Though this search technique has its disadvantages in Arabic, we allow using it in the search; a user might choose to apply it for a certain reason.

2- **Affixional Search option** By “Affix” it is meant, in this context, prefixes and\or suffixes adhered to a lexical item. The affixional technique can be called “Intelligent wildcard Search” opposite to the “dummy wildcard search”. The dummy search accepts any characters adhered to the right or the left of the query word (assuming that these characters are just prefixes or suffixes, which is right in relation to the English language). Applying such technique on Arabic documents may result in retrieving irrelevant words; queries like: **عُمَّال, ساخر ,عبير** , may include in the search results, words like: **استعمال , مساخر , تعبير** respectively.
According to the intelligent wildcard search, valid affixes for each query word are pre-defined, thus; the query word with all its affixional variations is only included in the search. i.e. the affixional search

technique automatically includes in the search process, the query word and all its possible valid variations. This technique covers what is missed in the Exact Search. It is suitable for searching proper nouns in Arabic.

Inflectional Option

In Arabic, some of the functional meanings - that are expressed through affixes added to the lexical item in European languages- are expressed in Arabic through internal change in the body of the lexical item: **كتاب → كتب , حرب → حروب , ولد → أولاد , شباك → شبابيك , عاصفة → عواصف**

In Arabic, the irregular plural is the most common type of inflectional issues. It is extensively used; it applies on more than one third of common nouns, thus; it cannot be ignored as a linguistic phenomenon, meanwhile, broken plurals cannot be registered in a list as being done in English.

The inflectional search technique is designed to overcome issues like that of irregular plural and irregular feminine: **أصغر → صغرى , أحمر → حمراء.** It covers the variations of a query word that are not expressed through affixes.

This search option enhances searching on common nouns.

Writing Options

In Arabic there are two writing systems judged as correct. We call them "Strict" and "Relaxed" modes. The strict mode applies the classical writing rules; between them is the differentiation among the four graphemes " أ, إ , آ ,ا ", and the two graphemes " ي , ى ". In modern standard Arabic some modifications are introduced to the writing system and considered as acceptable. These modifications are of geographical character. Among these modifications is: dropping Hamza " ء " and using the grapheme "ا " to represent " أ ", " إ ", " ا " (**أكرم , إقبال, اكتب → اكرم , اقبال , اكتب**) and dropping the final yaa: ي , thus, the grapheme " ى " represents both " ى & ي " (**على , علي → على , على**). Among these modifications are the set of words like **مسئول** that is written as **مسؤول.**

The writing system that considers these modifications is what we call "relaxed mode". It is to note that modifications in the relaxed mode go in one direction, i.e. the opposite replacements are not allowed: neither أ or إ can replace the two other graphemes: ي as well cannot replace **ى.**

On the web, the writing errors dominate in most of the Arabic contents. To insist on ignoring wrong words, and considering only the correct ones in both indexing and searching processes, is not a good idea. Ignoring incorrect words means dropping a good quantity of data that express concepts.

the fact that search targets concepts and not words, a third writing option was added to the writing modes; namely; "the common error mode".

The Analysis of a big corpus full of writing errors, helped with recognizing the most common word errors. They were classified and arranged according to certain priorities. Some error types are introduced to the common error mode: Hamazaat (**أبراهيم , إستقبال**), Yaa' (**مستشفي**), Taa' and Haa' (**أقلامة , حديقه**).

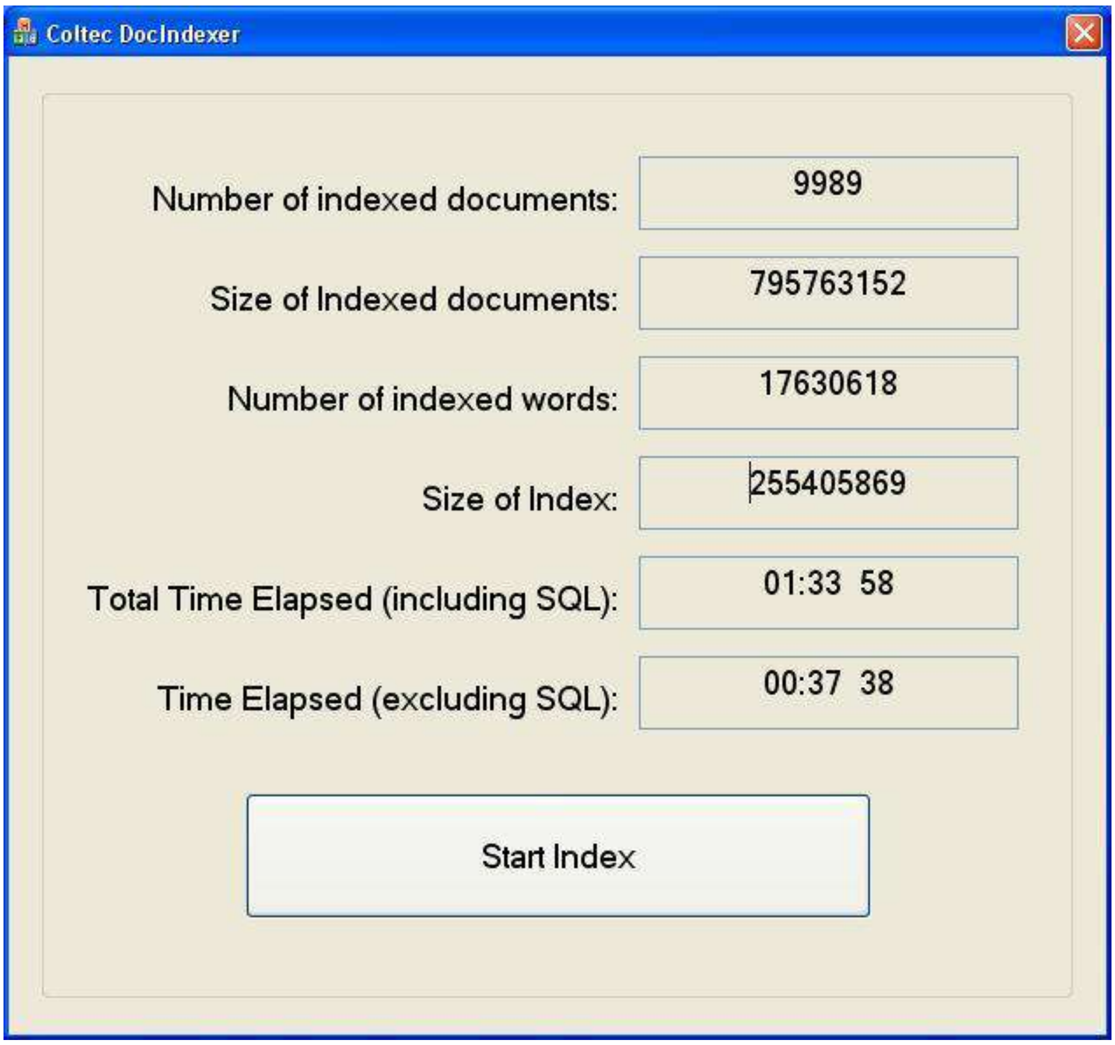
This mode is the default in the search process. The user can choose one of one of the two other options "strict" or "relaxed" if he is dealing with linguistically revised sites.

It worth mentioning, that the common error mode has its disadvantages that require more investigation. The main issue is that a wrong word might be mixed with a similar right word: if the user means "after it" "**عقبه** "and writes the last character wrongly as Taa' Marbuuta " **عقبة** ", this wrong word mixes with a right one whose meaning is "obstacle".

<u>Ambiguity Solution</u>

A temporary solution is being applied; the affixes specific to each of the stop words is removed. What remains from the affixes of each stop word are those valid for similar full words: " **عليك، عليهم، عليه**" are removed, but “على ” remains because it might point to the proper noun “ عَلَىّ “. This way, we get rid of a part of redundancies in search results. The opposite is done to guarantee the search comprehensiveness in case of semantic ambiguity; we create what we call “shell affixation system”; it includes all affixes of semantically ambiguous words in addition to their inflectional forms in the search operation. If the query word is “**منازل** ”, then “ **منازلان، منازلون...** ” in addition to "**منزل، منزلان...**" will be included in the search results.

All of the mentioned NLP facilities are exercised during the indexing phase. The linguistic information related to each indexed item, is saved with it. Thus, no extra process has to be applied in the querying phase which guarantees a high search speed. Meanwhile, the indexing process is not highly affected by added NLP work; one reason is the proper structure of primary linguistic data, the other reason is that the index design is dedicated to deal with the required NLP tasks.
Following is a screen shot assigning information on the index process as being monitored by a simple monitoring tool built specifically for follow up purposes:


Coltec DocIndexer

Number of indexed documents:	9989
Size of Indexed documents:	795763152
Number of indexed words:	17630618
Size of Index:	255405869
Total Time Elapsed (including SQL):	01:33 58
Time Elapsed (excluding SQL):	00:37 38

The in use PC is with the following specs:
Intel Core2 Duo 1.80GHz, 2MB cache
2GB RAM

MS Windows XP Professional sp3
MS SQL Server 2008 Enterprise RTM

It is to note that

1- The index size is about one third of the size of indexed documents
2- The indexing rate is 290774 words/minute (storage time of the SQL server being included)
3- The indexing rate raises to 468484 words/ minute when the storage time of the SQL server being excluded

VI. Conclusions

In this paper, we reviewed the specific issues of the Arabic language that cannot be ignored or handled superficially. We showed the necessity of introducing NLP techniques in Arabic search engines to overcome a part of the linguistic distortion that affects both indexing and searching.
Affixional and inflectional issues are almost overcome through the NLP based search options. Yet, we have ideas to improve the applied techniques.
The writing issues are partially solved through the options of writing modes. More investigation has to be done to control mixing up correct words with misspelled ones in the common error mode.
The linguistic ambiguity is a serious issue. It requires more time and efforts to be fundamentally solved.

VII. References

[1] E.M. Voorhees and D. Harman:TREC; Experiment and Evaluation in Information Retrieval, MIT Press, 2005

[2] H. Abu-Salem, M. Al-Omari, and M.Evens, Stemming methodologies over individual query words for Arabic information retrieval. JASIS, 50 (6), pp. 524-529, 1999

[3] J.I. Tait, Charting a New Course: Natural Language Processing and Information Retrieval: Essays in Honour of Karen Spärck Jones, Springer, 2005.

[4] J. Xu, A.Fraser, and R.Weischedel, Empirical studies in strategies for Arabic retrieval. Sigir Tampere, Finland: ACM, 2002.

[5] L.S. Larkey & M.E.Connell, Arabic information retrieval at UMass in TREC-10., TREC 2001, Gaithersburg: NIST, 2001.

[6] L.S. Larkey, L. Ballesteros, and M.E. Connell, Improving stemming for Arabic information retrieval: Light stemming and co-occurrence analysis, University of Massachusetts, Technical Report IR-249, 2002

[7] M. Piotrowski, NLP-supported full-text retrieval, CLUE technical report no. 3, Friedrich-Alexander University Erlangen, 2001

[8] W.Kraaij, & R.Pohlmann, Viewing stemming as recall enhancement. In Proceedings of ACM SIGIR96. pp. 40-48, 1996.

[9] TREC homepage: http://trec.nist.gov/overview.html

[10] NLP Track at TRECC-5 homepage:
http://www.dtic.mil/cgi-bin/GetRDoc?location-U2&doc= GetTRDoc.pdf&AD=ADA470634