Towards Diactritizing Arabic Text

Ramzi A. Haraty, Mohamad Mowafak Allaham, and Amer El-Homaissi
Department of Computer Science and Mathematics
Lebanese American University
Beirut, Lebanon
Email: rharaty@lau.edu.lb

Abstract

The Arabic language is one of the most complex languages that exist, yet one of the least supported languages in the computing world, which is causing its downfall. In this paper, we present a model to categorize Arabic words in order to figure out the function of each word in a sentence so it would be diactritized correctly. Experimental results show promising results.

Keywords: Arabic language, hidden Markov model, stemming, and diactritization.

1. INTRODUCTION

The Arabic sentence can be of two forms: a sentence where the placement of the verb is at the beginning of the sentence, and a sentence where the placement of the noun is at the beginning of the sentence [1]. In each of these sentences, different combinations of nouns can take place where the decision on if these sentences make sense or not is easy for a human mind to do. However, for a computer, to make such a decision is not something easy. How can we make a computer understand that an apple cannot eat a child? This is the biggest challenge of this work. Understanding a sentence and how the relationship between nouns changes according to a verb, are a must for any intelligence to be able to diactritize an Arabic sentence correctly. What is the role of each noun in a sentence? What is the best strategy to be followed to reach the highest success rate? We answer these questions in our proposed work. We believe our work can be used to a variety of contexts including, automatically diactritizing books released – they will not have to be diactritized manually by a human. Our work can also be used to diactritize newspapers, which is important for people to read Arabic correctly, and for new learners to practice their Arabic.

In this paper, we present a model to categorize Arabic words in order for the program to be able to figure out the function of each word in a sentence so it would be diactritized correctly. This model will help people, schools of Arabic language and students to better learn the Arabic language using computers. It will help news agencies to make their news more readable, especially for beginners of the Arabic language. It will also help writers and publishing houses from a grammatical perspective to accelerate the release of books.

The rest of the paper is organized as follows. Section 2 presents related work. Section 3 presents our model. Experimental results are discussed in section 4. And section 5 concludes the paper and presents future work.

2. LITERATURE REVIEW

There have been a number of efforts that deal with the Arabic language. In [2], the authors discuss a Part-Of-Speech (POS) tagger for Arabic. The paper presents the characteristics of the Arabic language and the POS tag set that was selected, and outlines the details of their development methodology using the Hidden Markov Model (HMM) for Arabic. The authors claim that the HMM POS tagger was 97% accurate.

In [3], the authors investigate the proportion of vocalic intervals and the standard deviation of consonantal intervals in six dialects. Their results show that the main factors responsible for differences in rhythmic structures are complex syllable and reduced vowels in the western dialects, and longer vowels in the eastern dialects. The paper also discusses discrete or continuous natures of rhythm types.

The authors of [4] discuss the information retrieval problem of auto-indexing Arabic documents. Auto-indexing a text document refers to automatically extracting words that are suitable for building an index for the document. The authors also propose an auto-indexing method for Arabic text documents. The method is mainly based on morphological analysis and on a technique for assigning weights to words. The morphological analysis uses a number of grammatical rules to extract stem words that become candidate index words. The weight assignment technique computes weights for these words relative to the container document. The weight is based on how spread is the word in a document and not only on its rate of select the more important index words.

In [5], the authors present some of the initial findings in the development of an Arabic part-of-speech tagger. For this tagger, a tag set containing 131 tags (derived from traditional Arabic grammatical theory) is compiled. This tagger is used to manually tag a corpus and to extract a lexicon from this corpus.

A novel algorithm for restoring short vowels and additional diacritics, using a cascade of probabilistic finite-state transducers trained on the Arabic Treebank, is presented in [6]. The algorithm integrates a word-based language model. This combination of probabilistic methods and simple linguistic information yields important levels of accuracy.

In [7], a rules compiler that extends the Arabic indexing rules used by the indexer at runtime is proposed. An approach using a prototype that was built using Java and MS SQL Server as a backend RDBMS is proposed.

The authors of [8] discuss the techniques of stemming Arabic verbs and nouns and the analysis of words. This is important because it will assist the program to differentiate between verbs and nouns using stemming.

In [9], the authors show that Hidden Markov Models are a useful tool for the test of vowel restoration in Semitic languages. Their technique generalizes well to both Hebrew and Arabic. Using a publicly available version of the Bible and the Qur'an as corpora, a success rate of 86% for restoring the exact vowel pattern in Arabic and 81% in Hebrew is achieved.

In [10], the authors demonstrate the process of extracting temporal references from Arabic texts. Their algorithm is highly dependable on the stemming process. A list of all the temporal references is used; the type of the temporal word decides the procedure to treat this word and gives the importance of this temporal reference. The authors claim they received results of 95% precision and 91% recall.

3. THE PROPOSED MODEL

In order to let a machine understand Arabic sentences and process them, we have to find a method that combines both artificial intelligence and linguistics in order to reach our goal in diactritizing Arabic sentences. This method is manifested in the area of Natural Language Processing (NLP), which has numerous intersections between the two fields. A very core topic in the field of NLP is Part of Speech (POS) Tagging [11]. POS Tagging algorithms fall into two categories: rule based and stochastic [12]. Rule based taggers contain a large number of hand-crafted rules, whereas stochastic taggers used a tagged corps for training. The tagger implemented in our model, which we refer to as Shakkel, is based on the rule based approach. Our POS Tagger in Shakkel assigns tags to words based on the frequency of each word (for example, "الولد" is more often a "فاعل" than "التفاحة"). The frequency of a word is the number of times a word was repeated in Arabic texts.

3.1 Methodology

The sentence structure of an Arabic sentence is quite complicated. The location of words may vary differently in a sentence, ending up having the same meaning. Starting from the concept of E. Brill's Tagger [13], we implemented our own approach of tagging words in Arabic sentences, specifically (Al-Jumal Al-Fe'lyah, الجمل الفعلية). The implemented system mainly consists of three main components:

1. Database of words.
2. A tagger that tags words of a given sentence.
3. The Shakkel engine.

Our initial database design consists of two million words with their frequencies (Frequencies of words in Newspapers). The words are called corpora in the field of NLP. The Arabic corpus used in our project was downloaded from the University of Leeds [8]. In order to make use of this database, specifically in the process of Part of Speech tagging, we added several fields ending up having the following database design depicted in table 1.

Table 1. The database structure.

bigint [20]	bigint [20]	bigint [20]	bigint [20]	bigint [20]	bigint [20]	bigint [20]	bigint [20]	bigint [20]	varchar [20]	bigint [20]
Frqncy	Other	Place	Time	Sifaa	Object	Animal	Human	Verb	Word	ID
0	0	0	0	0	0	0	1	0	الولد	1
0	0	0	0	0	1	0	0	0	التفاحة	2

The fields (Other, Place, Time, Sifaa, Object, Animal, Human, and Verb) are used to simplify the tagging process, which will be explained in the "Procedure" section of this paper.
The initial value for the fields of an inserted word is zero except the category the user selects. For example, الولد is Human ➜ 1 (Increment the Human field) and التفاحة is Object ➜ 1 (Increment the Object field). Afterwards, the Shakkel engine will automatically tag الولد as HMN and التفاحة as OBJ and store them in a list. Since Shakkel is a contemporary project, no libraries or data structures were available to help in the development process. Consequently, it is implemented as an extensive reflex agent that works in a fully observable environment. Figure 1 displays the architecture of the Skakkel agent.

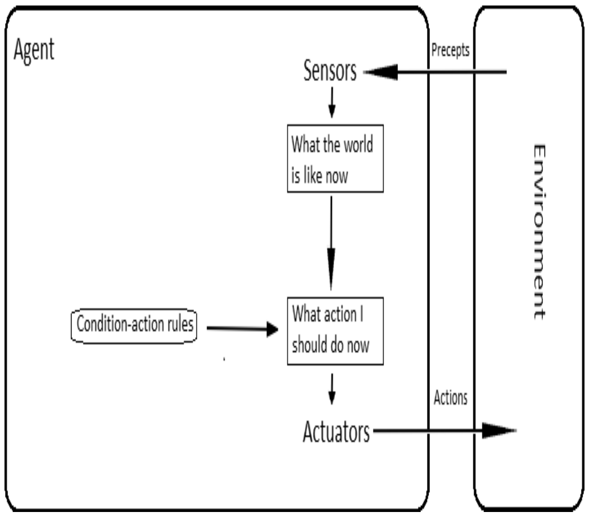


Figure 1. The abstract reflex agent.

The agent includes the following components:

- **Environment:** Arabic sentences (user input), fully observable.
- **Precepts:** Keyboard.
- **What the world is like now:** Defines the initial state of the software after a user finishes his/her input.
- **Condition-action rules:** Represents a set of grammatical rules of Al-Joumal Al-Feelyeh (الجمل الفعلية) in Arabic [13].
- **What action I should do now:** Is a stage where the agent diactritizes an inserted sentence based on the grammatical rules defined earlier.
- **Actuators:** Is the computer screen that will output the Arabic sentence after it has been diactritized.

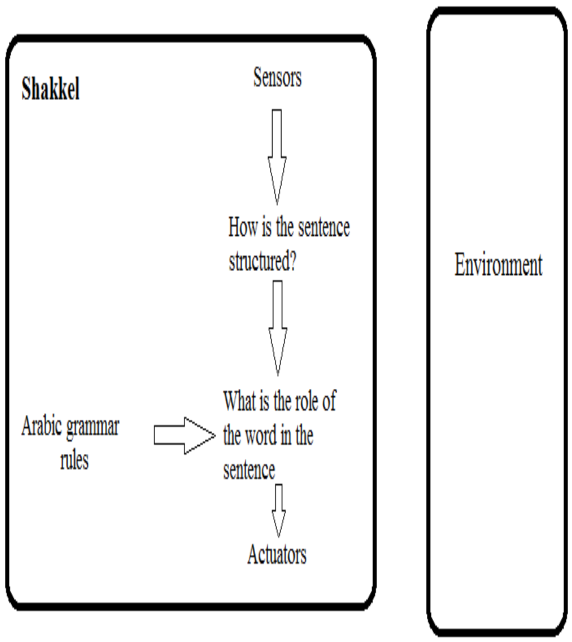
Figure 2 describes the Shakkel agent in details.

Shakkel uses reasoning in a very similar fashion to the one in forward chaining:

Sample input: أكل الولد في البيت

- **A**: في is a stop term (Haref Jar, حرف جر), (Tag: ST), (ST stands for Stop Term).
- **B**: البيت (Noun), (Tag: Unknown).
- **C**: البيت (Esem Majrour, اسم مجرور).

Conclusion ➔ البيتِis diactritized with (Kasrah, كسرة).


Shakkel
Sensors
How is the sentence structured?
Arabic grammar rules
What is the role of the word in the sentence
Actuators
Environment
Figure 2. The Shakkel engine.

Our agent has its own rule base, represented by a set of grammatical Arabic rules, which are traversed repeatedly for each word in a sentence until a premise can be concluded. Afterwards, a conclusion can be drawn, about the whole sentence, based on the premises that were set earlier for each word. After these two stages, a sentence will be fully diactritized.

3.2 The Process

The process of the diactritization of Arabic sentences in Shakkel is divided into three stages:

1. Analyzing user input.
2. POS (Part-of-Speech) Tagging.
3. Analysis of POS tags.

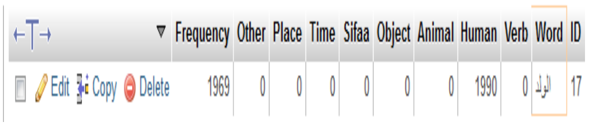
A user types in an Arabic sentence, for example a sentence that starts with a verb (Joumla Felyah, جملة فعلية). Afterwards the user input will be dissected into several strings in order to process each word separately. For example, أكل الولد التفاحة, will be tokenized into: ['أكل', 'الولد', 'التفاحة']. The first thing to check for is stop terms. A stop term is a term that does not really affect the meaning of a sentence (e.g., الذي - منذ-عند). In Shakkel, we defined special terms that affect the structure of a sentence. The following is a list of special terms recognized by Shakkel:

- حروف الجر: (في - من - إلى- عن - على - ل - ب)
- حروف العطف: (أو - و- لا - بل)
- أدوات الجزم و النهي: (لم – لا)
- أدوات النصب: (أن - لن – كي)

Part of speech tagging is a very important stage in the process of diactritizing Arabic texts. The accuracy of this stage will be reflected in the accuracy of the final result after a sentence has been diactritized. At first, Shakkel checks the tense of the verb available in the sentence and give the corresponding tag to it. For example,

- (Fel-Moudare, فعل مضارع) ➡ POS-Tag: **PRTV.**
- (Fel-Madi, فعل ماضي) ➡ POS-Tag: **PSTV.**
- (Fel-Moudare' Jari Lil Mostakbal, فعل مضارع للمستقبل) such as, سيدرس ➡ POS-Tag: **FTRV.**
- (Verbs of six letters, أفعال سداسية مضارعة) such as, يستخرج ➡ POS-Tag: **S_PRTV.**
- (Verbs of six letters, أفعال سداسية ماضية) such as, استخرج ➡ POS-Tag: **S_PSTV.**

Secondly, the database will be queried to return the maximum value in a tuple that corresponds to a specific word in the sentence. For example, take the sentence أكل الولد التفاحة. The screenshot of the tuple "الولد" is depicted in figure 3. The agent will return "1990" since it is the maximum of all values. Note that Frequency and Verb fields are not included in the comparison; they are only used for research purposes. Since Human is the field which contains the maximum value, then the word "الولد" will be tagged as "HMN".



	Frequency	Other	Place	Time	Sifaa	Object	Animal	Human	Verb	Word	ID
Edit Copy Delete	1969	0	0	0	0	0	0	1990	0	الولد	17

Figure 3. The "الولد" database tuple.

Table 2 presents a list of tags of all the fields in the database, and table 3 displays the tags for special terms used in our system.

In case a word is not in the database, a POS tag of "None" will be initially assigned to it until the entire sentence is tagged. Later, the list containing the POS tags of the words in a sentence will be processed again by a method and assigns a tag to this word, based on the formation of the sentence.

Table 2. POS tags representations.

Other	OTHR
Place	PLC
Time	TMaE
Sifaa	SFA
Animal	ANML
Human	HMN

Table 3. Tags for special terms [14].

حروف العطف	ATF
أدوات الجزم	JSM

Regarding special terms, like (Harouf Al-Jar, حروف الجار) there are special tags for them:

- (في - من - إلى- عن - على) ➡ **MJT**
- (ل) ➡ **LMJT**
- (ب) ➡ **BMJT**

Any other stop terms other than the ones mentioned above will have **(ST)** as a tag for it. Finally, a list containing part of speech tags assigned to words in a sentence will be used to diactritize the sentence. The experimental results of Shakkel are presented in the next section.

4. EXPERIMENTAL RESULTS

We implemented our work using a Hidden Markov Model and a rule based approach. We used the Python programming language for implementation, because Python has a rich set of lists capabilities. We also used the Arabic corpora of the University of Leeds as our database. We did not find the results of previous work on diacritizing Arabic text references to compare with the result of our work. However, we can show promising results of diacritizing words in Arabic texts.

We used the following list of tags for our experimentation. For a complete listing, interested readers are referred to [10].

<u>Verbs:</u>

PSTV:	فعل ماضي
PRTV:	فعل مضارع
FTRV:	فعل مستمر في الحاضر و ذاهب للمستقبل
S_PRTV:	فعل مضارع سداسي
S_PSTV:	فعل ماضي سداسي
AMR_V:	فعل أمر
N_PRTV:	فعل مضارع منصوب

<u>Stop Terms:</u>

JAZM:	أدوات الجزم
NSBT:	أدوات النصب
ST:	كلمات لا توئثر على تشكيل الجملة، مثال: عند
MJT:	حرف جر
DMJT:	منذ، أداة جر مرفوعة بالضمة
BMJT:	حرف الجر الباء

LMJT: حرف الجر اللام
ATF: حروف العطف
F_ATF: حرف العطف الفاء

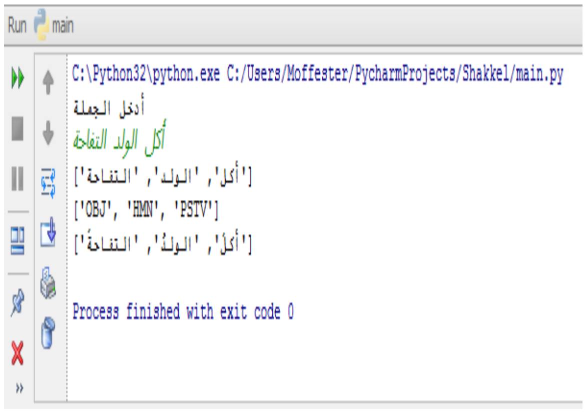
Database Tags:

HMN:	كلمة تستخدم للإنسان
OBJ:	كلمة تستخدم للجماد
PLC:	تستخدم للمكان
V:	تستخدم للأفعال
ANM:	تستخدم للحيوان
SFA:	تستخدم للنعت
TME:	تستخدم للوقت

Other Tags:

MATOUF:	للاسم المعطوف
ESMJR:	اسم مجرور

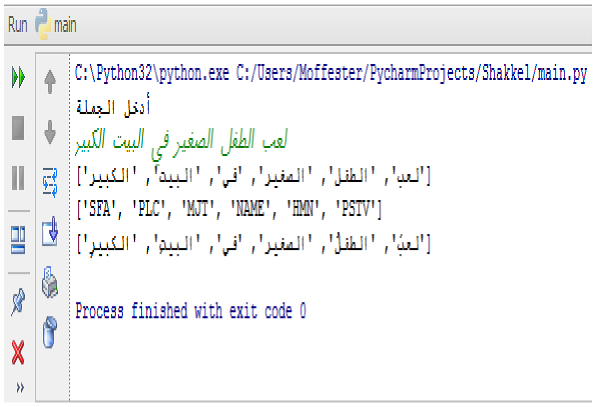
The following shots (figures 4 and 5) show some sample outputs of Shakkel.



```
C:\Python32\python.exe C:/Users/Moffester/PycharmProjects/Shakkel/main.py
أدخل الجملة
أكل الولد التفاحة
['أكل', 'الولد', 'التفاحة']
['OBJ', 'HMN', 'PSTV']
['أكلَ', 'الولدُ', 'التفاحةَ']

Process finished with exit code 0
```

Figure 4. Screenshot of Shakkel output result (1).



```
C:\Python32\python.exe C:/Users/Moffester/PycharmProjects/Shakkel/main.py
أدخل الجملة
لعب الطفل الصغير في البيت الكبير
['لعب', 'الطفل', 'الصغير', 'في', 'البيت', 'الكبير']
['SFA', 'PLC', 'MJT', 'NAME', 'HMN', 'PSTV']
['لعبَ', 'الطفلُ', 'الصغير', 'في', 'البيتِ', 'الكبيرِ']

Process finished with exit code 0
```

Figure 5. Screenshot of Shakkel output result (2).

Several other examples are shown below:

Example 1 (نعت ، حرف جر، اسم المجرور):
- Original Sentence: ['يلعب', 'الطفل', 'الكبير', 'الجميل', 'في', 'المدرسة', 'الصغيرة'']
- POS Tagging: ['SFA', 'PLC', 'MJT', 'SFA', 'SFA', 'HMN', 'PRTV']
- Final result: ['يلعبُ', 'الطفلُ', 'الكبيرُ', 'الجميلُ', 'في', 'المدرسةِ', 'الصغيرةِ'']

Example 2 (حروف الجر المتصلة بالكلمة):
- Original Sentence: ['ذهب', 'الولد', 'للعب']
- POS Tagging: ['LMJT', 'HMN', 'PSTV']
- Final result: ['ذهبَ', 'الولدُ', 'للعبِ']

Example 3 (الأفعال السداسية):
- Original Sentence: ['يستخرج', 'العامل', 'الذهب']
- POS Tagging: ['OBJ', 'HMN', 'S_PRTV']
- Final result: ['يستخرجُ', 'العاملُ', 'الذهبَ']

Example 4 (حروف العطف):
- Original Sentence: ['سافر', 'موفق', 'و', 'عامر']
- POS Tagging: ['MATOUF', 'ATF', 'HMN', 'PSTV']
- Final result: ['سافرَ', 'موفقُ', 'و', 'عامرُ']

Example 5 (أداة الجزم لا، حرف العطف متصل بالكلمة):
- Original Sentence: ['لا', 'تقصر', 'فتندم']
- POS Tagging: ['F_MATOUF', 'V_MAJZM', 'JAZM']
- Final result: ['لا', 'تقصرْ', 'فتندمْ']

Example 6 (حروف العطف):
- Original Sentence: ['سافر', 'زهير', 'لا', 'خالد']
- POS Tagging: ['MATOUF', 'ATF', 'HMN', 'PSTV']
- Final result: ['سافرَ', 'زهيرُ', 'لا', 'خالدُ']

Example 7 (فاء السببية في محل عطف):
- Original Sentence: ['دخل', 'موفق', 'فسامر']
- POS Tagging: ['F_MATOUF', 'HMN', 'PSTV']
- Final Result: ['دخلَ', 'موفقُ', 'فسامرُ']

Example 8 (أداة جزم و نهي لم):
- Original Sentence: ['لم', 'نقرأ', 'الدرس', 'في', 'المدرسة']
- POS Tagging: ['PLC', 'MJT', 'OBJ', 'V_MAJZM', 'JAZM']
- Final Result: ['لم', 'نقرأْ', 'الدرسَ', 'في', 'المدرسةِ']

Example 9 (لن، أداة نصب تنصب الفعل المضارع):
- Original Sentence: ['لن', 'نلعب', 'بالكرة', 'في', 'المنزل']
- POS Tagging: ['PLC', 'MJT', 'BMJT', 'N_PRTV', 'NSBT']
- Final Result: ['لن', 'نلعبَ', 'بالكرةِ', 'في', 'المنزل']

5. CONCLUSION

In this work we presented a model for diactritizing Arabic text. The model demonstrates the process of extracting the word from texts. This model is highly dependable on the stemming process. A list of all the tags was used to aid in identifying words. The type of the word decides the procedure to treat this word and gives it its proper diactritization. Experimental results show promising results.

Future work plans include adding even more grammar rules to Shakkel, and streamline strategies to increase its accuracy and reliability.

6. REFERENCES

[1] Schulz, Eckehard: *Standard Arabic: An Elementary - Intermediate Course*. (ISBN13: 9780521774659), Cambridge University Press (2000).

[2] Al Shamsi, Fatma and Guessoum, Ahmed: *A Hidden Markov Model–Based Part of Speech Tagger for Arabic.* Proceeding of the 8th International Conference on the Statistical Analysis of Textual Data, France, pp. 31–42, (2006).

[3] Ghazali, Salem, Hamdi, Rym, and Barakat, Melissa: *Speech Rhythm Variation in Arabic Dialects.* Proceedings of Prosody 2002, Aix-en-Provence, pp. 331-334, (2002).

[4] Mansour, Nashat, Haraty, Ramzi A., Daher, Walid, and Houri, Manal: *An Auto-indexing Method for Arabic Text.* Information Processing and Management Journal, Volume 44, Issue 4, pp. 1538–1545, (July 2008).

[5] Khoja, Shereen: *APT: Arabic Part-of-speech Tagger.* Proceedings of HLT-NAACL: Short Papers, pp. 149-152, (2004).

[6] Nelken, Rani and Shieber, Stuart: *Arabic Diacritization Using Weighted Finite-state Transducers*. Proceedings of the ACL-05 Workshop on Computational Approaches to Semitic Languages, pp. 79--86, Ann Arbor, Michigan (2005).

[7] Raheel, Saeed M: *Extensibility in Arabic Full-Text Indexing*. M.S. Thesis, Department of Computer Science and Mathematics – Lebanese American University, (2003).

[8] Latifa Al-Sulaiti: *Arabic Corpora,* University of Leeds, Department of Computer Science. Retrieved on March 1, 2013 from http://www.comp.leeds.ac.uk/eric/latifa/arabic_corpora.htm.

[9] Yakov, Gal: *A Hidden Markov Model Approach to Vowel Restoration in Arabic and Hebrew*. Proceedings of the ACL-02 Workshop on Computational Approaches to Semitic Languages, pp. 1-7, (2002).

[10] Haraty, Ramzi A., and Khatib, Samer: *TREX: A Temporal Reference Extractor for Arabic Texts*. Proceedings of the Third ACS/IEEE International Conference on Computer Systems and Applications. Cairo, Egypt, (2005).

[11] Toutanova, Kristina and Manning, Christopher: *Enriching the Knowledge Sources Used in a Maximum Entropy Part-of-Speech Tagger.* Proceedings of the Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora (EMNLP/VLC-2000), pp. 63-70, (2000).

[12] van Halteren, Hans, Zavrel, Jakub, and Daelemans, Walter: *Improving Accuracy in NLP Through Combination of Machine Learning Systems*. Computational Linguistics, 27(2), pp. 199–229, (2001).

[13] Brill, Eric: *A Simple Rule-based Part of Speech Tagger*. Proceedings of the third conference on Applied Natural Language Processing (ANLC '92). Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 152-155, DOI=10.3115/974499.974526, (1992).

[14] Wikipedia, *حروف العطف*. Retrieved on March 1, 2013 from http://ar.wikipedia.org/wiki/.

BIOGRAPHIES

Ramzi A. Haraty is an associate professor of Computer Science in the Department of Computer Science and Mathematics at the Lebanese American University in Beirut, Lebanon. He is also the academic and internship coordinator for Middle East Program Initiative’s Tomorrow Leader’s program. He received his B.S. and M.S. degrees in Computer Science from Minnesota State University - Mankato, Minnesota, and his Ph.D. in Computer Science from North Dakota State University - Fargo, North Dakota. His research interests include database management systems, artificial intelligence, and multilevel secure systems engineering. He has well over 110 books, book chapters, journal and conference paper publications. He supervised over 110 dissertations, theses and capstone projects. He is a member of the Association of Computing Machinery, Institute of Electronics, Information and Communication Engineers, and the International Society for Computers and Their Applications.

Mohamad Mowafak Allham and **Amer El-Homaissi** received their B.S. degrees in Computer Science from the Lebanese American University. Their research interests include computer forensics and Arabic text processing.