Computational Methods to Vocalize Arabic Texts

Hani Safadi[1], Dr. Oumayma Dakkak[2], Dr. Nada Ghneim[2]
hanisaf@gmail.com, odakkak@hiast.edu.sy, n_ghneim@netcourrier.com
Faculty of Informatics Engineering, University of Damascus, SYRIA[1]
Higher Intuition of Applied Science and Technology, Damascus, SYRIA[2]

Abstract

Arabic Language has two kinds of vowels: Long vowels which are written as normal letters; and short vowels which are written as punctuation marks, above or below letters. Those short vowels are normally omitted in Arabic texts because the reader can fill them and guess the meaning based on his knowledge of the language, and the context in which the words are.
However, with the widespread usage of computers in linguistics application; Arabic texts need to be supplied with short vowels in order to be analyzed. Search engines, text to speech engines, and text mining tools are just some examples of applications that need Arabic texts to be vocalized before being processed.
In this paper, we present a new method to supply those vocals. The approach emphasizes on unsupervised machine methods, because public Arabic corpora are not available. Arabic rich morphology and diverse orthography present serious challenges for this approach. An algorithm is developed and a system is implemented in Java.
The techniques presented in this dissertation can be applied to similar Semitic or other languages that have the same problem.

1. Introduction

There are two types of vowels in Arabic: long vowels /A/, /w/, /y/ (We are using here Buckwalter Arabic trasliration table [6] see Appendix A) (although /w/, /y/ can be used as consonant sometimes), and short vowels /a/ (Fatha), /u/ (Damma), /I/ (Kasra). These short vowels are part of the word and are written as additional marks above or below letters. In addition to the short vowels, there are other marks (F Tanween-Fateh pronounced /an/, N Tanween-Damm pronounced /un/ or /on/, K Tanween-Kasir, pronounced /in/ or /en/, o Sukun which means that the consonant is not followed by a vowel), ~ Shadda which means a duplication of the consonant). All these marks are usually not written because Arabic reader can guess them, based on his knowledge of the language and on the context. They are only put when very necessary, in cases where the word is so ambiguous without them, and cannot be distinguished from other words. Even with such a situation, only one or two discriminating letters are vocalized.

For non-Arabic speakers this problem might seem too abstract. To illustrate it we will use a hypothetical version of English, we will call it AEng. In AEng, of the five vowels which are /a/, /e/, /i/, /o/ and /u/, we only write the /i/ and /u/ as normal letters, while we omit the /a/, /o/ and /e/.
Now try to read the following sentence in this language:
"If yu cn rd this thn w cn omt vwls"
If you figured out the meaning of this sentence then you are convinced that some vowels can be omitted. If not, you will adapt to this as you use AEng is your daily life!
The original sentence in English is:
"If you can read this then, we can omit vowels"
It is a little bit similar to chatting language.
Note that this is only an example, and does not need to be fully representative.
From this example, we see how ambiguous a sentence is without the short vowels (at least for computers). For example, the word Elm in Arabic could be:
Eilom Science
Ealam Flag
Ealim to know
There are also other possibilities….
In AEng we have the word bg that could be read or beg or bag.
This example makes it clear that we need short vowels in order to understand the text in computer.
Each letter has one or two short vowels associated with it.
The short vowels on all characters, but the last one, differentiate the word from other words. The short vowel on the last letter, however, determines the syntactic position of the word in the sentence (Like Nominative, Accusative, and Dative in German).
Although they add other information to the word, this study will not deal with them. In fact, putting the last vowels of the words needs deeper syntactic analysis, and the information they add are not so essential.
For example:
EilomF
EilomN
EilomK

2. Previous Attempts

Despite the abundance of computational Arabic studies, Arabic Vocalization is not enough studied.
Sakhr [1] has a commercial system for Arabic vocalization. Unfortunately, the system is totally closed and no information about it is known.
Y. Gal [2] uses a hidden Markov model (HMM) trained on vocalized Arabic texts. It uses the Holy Quranic text as a training corpus. It shows results of 85% correct vocalization when used with text from the same corpus.
R. Nelken and S. Shieber [3] use weighted finite-state transducers trained on LDC corpus [4] of M. Maamouri et al. The reported results show 93% of correct vocalization.

3. Our Methodology

While the past two approaches provide useful attempts to solve the problem; both of them have the same drawback: They tackle the problem with a top-down approach, building a model and training it with a corpus, although [3] did some morphological analysis.
The problem with this approach is that it is highly dependent on the corpus. For example, Quranic texts used in [2] are not good representative of modern Arabic. And newspaper archives in LDC [4] used in [3] do not cover all the topics in the language.
Our work uses a bottom-up approach, where we do a complete linguistic analysis of Arabic texts.
Our systems works in four stages:

1. Parsing the text.
2. Analyzing the text morphologically.
3. Part of speech tagging of the text.
4. Applying linguistic heuristics rules.

We will explain each of these steps, and show how it participates in the vocalization process.

1. Parsing: In this step, the text is spitted to phrases and each phrase is splitted to words. The process is simple; it uses the regular expressions to parse the text.
2. Morphological Analysis: In this step, each word is considered alone and passed to the morphological analyzer. The morphological analyzer provides for each word all the vocal possibilities which can be added to the word to generate words in the language. This stage gives also the part of speech tag of each possibility.
   We used Buckwalter Arabic morphological analyzer [5] that uses a concatinative methodology to analyze words. It considers each word as a combination of prefix, stem, and suffix. It has a dictionary of prefixes, stems, and suffixes, and two compatibility tables: prefix-stem table which tells what prefixes can come with what stems, and stem-suffixes table which tells what stems can come with what suffixes.
   All the possible combinations of a word to prefix-stem-suffix are considered (as long as the stem length is not zero). For each combination, the prefix, stem and suffix are checked whether they are contained in the dictionaries or not. If so, the compatibility between them is considered; if they are compatible, this combination is considered as a possible analysis to the word.
3. Part of speech tagging: After morphologically analyzing each word, we get the several possibilities of the word; each possibility has a corresponding vocalization, and part of speech. To choose the correct vocalization, we must choose the correct part of speech (POS).
   We built a POS tagger for Arabic, using unsupervised transformational based learning methods [7, 8]. We used the unsupervised methods because we did not have a tagged Arabic corpus. The only tagged Arabic corpus is LDC Arabic Corpus [4] that we could not afford to buy it. The tagger is trained using collected texts from Internet covering multiple disciplines [9].
   The generated rules are then examined manually. For some words, the part of speech tagging cannot resolve the ambiguity; therefore we need an additional level of processing.
4. Heuristic rules of disambiguation: These rules were met by language experts. The rules specify the choice of a certain part of speech with regards to the word and its context. For example, one of these rules specify: "if the word length is less than 3 letters, and one of its part of speech is preposition, then choose it".
5. Finally, if after all these levels, the ambiguity still remains in the word; a random choice of the part of speech tagged is made.

4. Implementation

We implemented the system using Java™ programming language. The implementation itself is composed of the same parts that are described previously, plus some additional tools and graphical user interfaces.
The implementation allows the user to use each part alone, or use all of them together.

5. Results

Unfortunately, because of the lack of vocalized Arabic texts, we did not have the opportunity to test our system thorough fully. However, we did some empirical tests. We vocalized large Arabic texts automatically and gave the results to experts in order to evaluate them. The evaluation shows a percentage of 80-90% of correct vocalization.

6. Future Works

Although our system uses a detailed analysis of texts, there are still some missing steps of analysis, these steps will improve the performance of our system. These steps may include:

1. Syntactic Analysis: Our current system cannot restore vowels on the last character of each word, because these vowels are determined according to the word position in the phrase (Subject, Object, Complement ...). Although texts without these last vowels are well understood, adding this additional level will certainly improve the performance.
2. Semantic Analysis: Doing a 100% Vocalization of an Arabic text requires a full understanding of the text. Although this is not at all easy for computers, some semantic analysis can help improving the performance. For instance, of the types of errors we still have in the current system, is when a word has several possibilities and all the possibilities have the same part of speech. Without semantic analysis we cannot solve this type of problems.
3. Pragmatic Analysis: This type of analysis is useful in conversations and idiomatic expressions.

Other improvements include enhancing the current modules, especially the part of speech tagger. We mentioned that we used unsupervised methods because we did not have a tagged corpus. We think that using a supervised method, trained on a tagged corpus, can significantly enhance the part of speech tagging performance.

Our work is not finished and we are still working on improving it.

7. Conclusion

The problem of restoring vocals in Arabic is essential for computational applications. Some attempts were made. We have provided a solution based on linguistic analysis. An implementation is done, and the results are promising. We are planning to improve and enhance the system.

8. References

[1] Sakhr Website http://www.sakhr.com/Sakhr_e/Technology/Diacritization.htm

[2] Ya'akov Gal, “An HMM Approach to Vowel Restoration in Arabic and Hebrew”, ACL 02 Semitic Language Workshop, 2002.

[3] Rani Nelken and Stuart M. Shieber, “Arabic Diacritization Using Weighted Finite-State Transducers”, Proceedings of the 2005 ACL Workshop on Computational Approaches to Semitic Languages, pages 79-86, Ann Arbor, Michigan, June 2005.

[4] Mohamed Maamouri, Ann Bies, Tim Buckwalter, and Wigdan Mekki, “The Penn Arabic Treebank: Building a Large-Scale Annotated Arabic Corpus”, Paper presented at the NEMLAR International Conference on Arabic Language Resources and Tools, Cairo, Sept. 22-23, 2004.

[5] Buckwalter T., “Buckwalter Arabic Morphological Analyzer Version 1.0”. Linguistic Data Consortium, catalog number LDC2002L49, ISBN 1-58563-257-0, 2002.

[6] Buckwalter T. Website, Arabic transliteration page www.qamus.org/transliteration.htm

[7] Daniel Jurafsky and James H. Martin, "Speech and Language Processing", P 288- 300, Prentice Hall, 2000.

[8] Eric Brill, “Unsupervised Learning of Disambiguation Rules for Part of Speech Tagging”, Proceedings of the Third Workshop on Very Large Corpora, P 1 – 13, Somerset, New Jersey, 1995.

[9] From The Global Arabic Encyclopedia website, http://www.mawsoah.net/

Appendix A
Buckwalter Arabic Transliteration Table [6]:

Transliteration	Unicode Value and Unicode Name
'	U+0621 ARABIC LETTER HAMZA
|	U+0622 ARABIC LETTER ALEF WITH MADDA ABOVE
>	U+0623 ARABIC LETTER ALEF WITH HAMZA ABOVE
&	U+0624 ARABIC LETTER WAW WITH HAMZA ABOVE
<	U+0625 ARABIC LETTER ALEF WITH HAMZA BELOW
}	U+0626 ARABIC LETTER YEH WITH HAMZA ABOVE
A	U+0627 ARABIC LETTER ALEF
b	U+0628 ARABIC LETTER BEH
p	U+0629 ARABIC LETTER TEH MARBUTA
t	U+062A ARABIC LETTER TEH
v	U+062B ARABIC LETTER THEH
j	U+062C ARABIC LETTER JEEM
H	U+062D ARABIC LETTER HAH
x	U+062E ARABIC LETTER KHAH
d	U+062F ARABIC LETTER DAL
*	U+0630 ARABIC LETTER THAL
r	U+0631 ARABIC LETTER REH
z	U+0632 ARABIC LETTER ZAIN
s	U+0633 ARABIC LETTER SEEN
$	U+0634 ARABIC LETTER SHEEN
S	U+0635 ARABIC LETTER SAD
D	U+0636 ARABIC LETTER DAD
T	U+0637 ARABIC LETTER TAH
Z	U+0638 ARABIC LETTER ZAH
E	U+0639 ARABIC LETTER AIN
g	U+063A ARABIC LETTER GHAIN
_	U+0640 ARABIC TATWEEL
f	U+0641 ARABIC LETTER FEH
q	U+0642 ARABIC LETTER QAF
k	U+0643 ARABIC LETTER KAF
l	U+0644 ARABIC LETTER LAM
m	U+0645 ARABIC LETTER MEEM
n	U+0646 ARABIC LETTER NOON
h	U+0647 ARABIC LETTER HEH
w	U+0648 ARABIC LETTER WAW
Y	U+0649 ARABIC LETTER ALEF MAKSURA
y	U+064A ARABIC LETTER YEH
F	U+064B ARABIC FATHATAN
N	U+064C ARABIC DAMMATAN
K	U+064D ARABIC KASRATAN
a	U+064E ARABIC FATHA
u	U+064F ARABIC DAMMA
i	U+0650 ARABIC KASRA
~	U+0651 ARABIC SHADDA
o	U+0652 ARABIC SUKUN
`	U+0670 ARABIC LETTER SUPERSCRIPT ALEF
{	U+0671 ARABIC LETTER ALEF WASLA
P	U+067E ARABIC LETTER PEH
J	U+0686 ARABIC LETTER TCHEH
V	U+06A4 ARABIC LETTER VEH
G	U+06AF ARABIC LETTER GAF