A Hybrid System for Automatic Arabic Diacritization

Mohsen A. A. Rashwan[1, 2], Mohammad Al-Badrashiny[1, 3], Mohamed Attia[1], Sherif M. Abdou[1, 4]
[1] The Engineering Company for the Development of Computer Systems; RDI, Egypt www.RDI-eg.com
[2] Prof. in the dept. of Electronics & Electrical Communications, Faculty of Eng., Cairo Univ., Egypt
[3] M.Sc. student in the dept. of Electronics & Electrical Communications, Faculty of Eng., Cairo Univ., Egypt
[4] Assistant Prof. in the Faculty of Computers & Information, Cairo Univ., Egypt
{Mohsen_Rashwan, Mohammed.Badrashiny, m_Atteya, sAbdou}@RDI-eg.com

Abstract

This paper introduces a two-layer stochastic system to automatically diacritize raw Arabic text that is known to be quite a tough problem. The first layer tries to decide about the most likely diacritics by choosing the sequence of full-form Arabic word diacritizations with maximum marginal probability via long A* lattice search and m-gram probability estimation. When full-form words happen to be out-of-vocabulary, the second layer is resorted to. This second layer factorizes each Arabic word into its possible morphological constituents (prefix, root, pattern and suffix), then uses m-gram probability estimation and A* lattice search to select among the possible factorizations to get the most likely diacritizations sequence. While the second layer has the advantage of excellent coverage over the Arabic language, the first layer enjoys a better disambiguation for the same size of training corpora especially for inferring syntactical (case-based) diacritics. The presented hybrid system enjoys the advantages of both layers. After a background on Arabic morphology & PoS tagging, the paper details the workings of both layers and the architecture of the hybrid system.

1. Introduction

Automatic words diacritization is one of the NLP challenges with languages having diacritics unveiling the phonetic transcription of their words. Arabic is an example of such languages where different diacritics over for the same spelling produce different words with maybe different meanings (e.g. عِلْم → “science”, عَلَم → “flag”, عَلَّمَ → “taught”, عَلِمَ → “knew” … etc.). Text-to-speech (TTS), Part-of-Speech (PoS) tagging, Word Sense Disambiguation (WSD), and Machine Translation can be enumerated among a longer list of applications that vitally benefit from automatic diacritization. One major challenge with Arabic is its rich derivative and inflective nature, so it is very difficult to build a complete vocabulary that cover all (or even most of) the Arabic generable words. In fact, while Arabic is on the extreme of richness as per its vocabulary when regarded as full-form words, this language is also on the extreme of compactness of atomic building entities due to its very systematic and rich derivative and inflective nature. Hence, the importance of sound Arabic language factorization, esp. morphological analysis, gets more crucial for Arabic diacritization.

Over more than 10 years we have been building Arabic diacritization systems that factorize input Arabic text into all the possible lexemes and case diacritics then statistically disambiguate the most likely sequence of these entities via deep lattice search, whence infer the most likely diacritization and phonetic transcription of the input text. While the virtue of this methodology is its excellent coverage of the language, its drawback is its relatively sluggish attenuation of the disambiguation error margin with increasing the annotated training corpora which is expensive and time consuming to build and validate.

So, we have started recently to try the same statistical language modeling and disambiguation methodologies over full-form Arabic words instead of factorized ones. While this proved to enhance the error margin faster than the former approach, it apparently suffers from the problem of poor coverage. It has then been realized that a hybrid of the two approaches may enjoy the best of each.

The architecture of this hybrid system is detailed over section 3 of this paper after revising our Arabic morphological analysis & PoS tagging models and the architecture of the Arabic diacritization disambiguating the factorized output after these analyses over section 2. Section 4 explains the statistical language modeling as well as lattice search deployed in both architectures. In section 5 the most recent related works are mentioned.

In section 6, the set-up and results of our experiments are described. Whence the paper is concluded over section 7 with suggestions for further future work.

2. Arabic Factorization via Morphological Analysis and PoS Tagging

The diacritization of an Arabic word consists of two components; morphology-dependent and syntax-dependent ones. While the morphological diacritization distinguishes different words with the same spelling from one another; e.g. عِلْم which means “science” and عَلَم which means “flag”, the syntactic case of the word within a given sentence; i.e. its role in the parsing tree of that sentence, determine the syntax-dependent diacritic of the word. For example; درسْتُ عِلْمَ الرياضيات implies the syntactic diacritic of the target word - which is an “object” in the parsing tree - is “Fatha”, while يفيدُ عِلْمُ الرياضياتِ جميعَ العلوم implies the syntactic diacritic of the target word – which is a “subject” in the parsing tree - is “Damma”.

2.1. Arabic Morphological Analysis:

Our Arabic morphological model assumes the canonical structure uniquely representing any given Arabic word *w* to be a quadruple of lexemes (or morphemes) so that $w \rightarrow q = (t: p, r, f, s)$ where *p* is prefix code, *r* is root code, *f* is pattern (or form) code, and *s* is suffix code. The type code *t* can signify words belonging to one of the following 4 classes: *Regular Derivative* (w_{rd}), *Irregular Derivative* (w_{id}), *Fixed* (w_f), or *Arabized* (w_a).

Prefixes and suffixes; *P* and *S*, the 4 classes applied on patterns giving F_{rd}, F_{id}, F_f, and F_a, plus only 3 classes applied on roots[1]; R_d, R_f, and R_a constitute together the 9 categories of lexemes in this model. The total number of lexemes of all these categories in our model is around 7,800. With such a limited set of lexemes, the dynamic coverage exceeds 99.8% measured on large Arabic text corpora excluding transliterated words. [5]

Table 1 below shows this model applied on few representative sample Arabic words.

Sample word	***Word type***	***Prefix & prefix code***	***Root & root code***	***Pattern & pattern code***	***Suffix & suffix code***
فَمَا	*Fixed*	فَــــ **2**	اَلَّذِي **87**	مَا **48**	- **0**
تَتَنَاوَله	*Regular Derivative*	تـــ **86**	ن و ل **4077**	تَفَاعَلَ **176**	ـه **8**
اَلْكِتَابَات	*Regular Derivative*	الـــ **9**	ك ت ب **3354**	فِعَال **684**	ـات **27**
اَلْعِلْمِيَّة	*Regular Derivative*	الـــ **9**	ع ل م **2754**	فِعْل **842**	ـيَّة **28**
مِنْ	*Fixed*	- **0**	مِـنْ **63**	مِـنْ **118**	- **0**
مَوَاضِيع	*Regular Derivative*	- **0**	و ض ع **4339**	مَفَاعِيل **93**	- **0**

Table 1: Arabic morphological analyses examples.

[1] The roots are common among both the regular and irregular derivative Arabic words.

2.2. Arabic PoS Tagging:

Our Arabic PoS-tagging model relies on a compact set of Arabic PoS tags containing only 62 tags that cover all the possible atomic context-free syntactic features of Arabic words. While many of these Arabic PoS tags may have corresponding ones in other languages, few do not have such counterparts and may be specific to the Arabic language.

This PoS tag-set has been extracted after a thorough scanning and redundancy elimination of the morpho-syntactic features of the 7,800 lexemes in our morphologically factorized Arabic lexicon. Completeness, atomicity, and insurability of these scanned morpho-syntactic features were the criteria adhered to during that process [2], [3].

Due to the atomicity of our Arabic PoS-tags as well as the compound nature of Arabic lexemes in general, the PoS labels of Arabic lexemes are represented by PoS tags-vectors. Each lexeme in our Arabic factorized lexicon is hence labeled by a PoS tags-vector.

While the Arabic PoS-tagging of stems is retrieved from the PoS label of the pattern lexeme only, not the root’s, the PoS-tagging of the affixes is obtained from the PoS labels of the prefix and suffix. So, the Arabic PoS-tagging of a quadruple corresponding to a morphologically factorized input Arabic word is given by the concatenation of its PoS labels of the prefix, the pattern, and suffix respectively after eliminating any redundancy. Table 2 shows the Arabic PoS-tagging of few sample words. For more details on this Arabic PoS-tagging model along with its underlying PoS tag-set refer to [3] and chapter 3 of [2].

Sample word	***Arabic PoS tags vector***
فَمَا	[Conjunction, Noun, Relative Pronoun, Null Suffix] [عطف، اسم، اسم موصول، لا لاحقة]
تَتَنَاوَله	[Present, Active, Verb ,Objective Pronoun] [مضارع، مبني للمعلوم، فعل، ضمير نصب]
اَلْكِتَابَات	[Definitive, Noun, Plural, Feminine] [ال التعريف، اسم، جمع، مؤنَّث]
مِنْ	[Null Prefix, Preposition, Null Suffix] [لا سابقة، حرف، لا لاحقة]
مَوَاضِيع	[Null Prefix, Noun, No SARF, Plural, Null Suffix] [لا سابقة، اسم، ممنوع من الصرف، جمع، لا لاحقة]

Table 2: Samples of PoS tag-vectors of Arabic words.

2.3. Arabic Diacritization via Statistically Disambiguating Factorized Arabic Text:

The morphological diacritization of a given word is directly extractable from the prefix, pattern, and suffix lexemes of the morphological analysis of that word. The issue here is to disambiguate the multiple analyses proposed by the Arabic morphological analyzer. In the absence of deeper linguistic processing, statistical disambiguation is deployed to infer the sequence of analyses with maximum likelihood probability according to a statistical language model built from a morphologically annotated training corpus.

For syntactic diacritization the PoS-tag vectors of a sequence of Arabic words along with the possible syntactic diacritics of each word are obtained after its morphological disambiguation. Statistical disambiguation is deployed again to infer the sequence of syntactic diacritics & PoS tags with maximum likelihood probability according to a statistical language model built from a training corpus annotated with PoS tags & syntactic diacritics [3]. Figure 1 shows the architecture of this diacritization system.

The deployed statistical disambiguation and language modeling [2], [4] in our diacritization system are described in section 4.

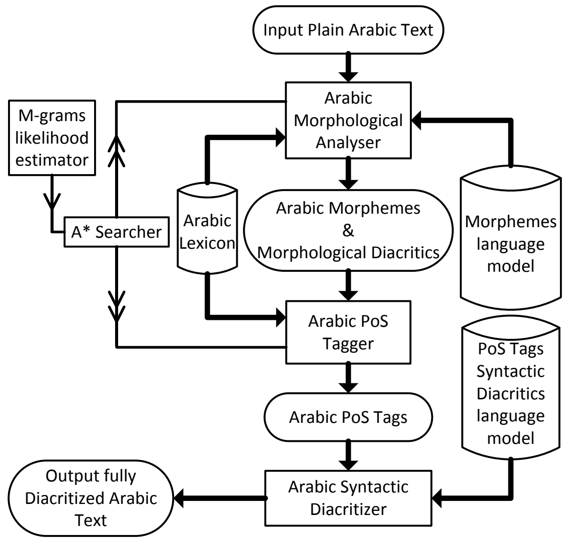
Input Plain Arabic Text
M-grams likelihood estimator
Arabic Morphological Analyser
A* Searcher
Arabic Lexicon
Arabic Morphemes & Morphological Diacritics
Morphemes language model
Arabic PoS Tagger
PoS Tags Syntactic Diacritics language model
Arabic PoS Tags
Output fully Diacritized Arabic Text
Arabic Syntactic Diacritizer

Figure 1: The architecture of Arabic diacritizer statistically disambiguating factorized Arabic text.

3. Disambiguating a Hybrid of Full-Form & Factorized Words

Aiming to enhance the performance of the Arabic diacritizer of factorized Arabic text we developed a hybrid system that combines the morphology based diacritizer with another diacritizer that is based on full-form words. Figure 2 shows the architecture of this hybrid Arabic diacritizer.

A large Arabic text corpus with a revised full morphological and syntactic phonetic annotation is used to build a dictionary of full-form Arabic words vocabulary. In the offline phase also, this text corpus is indexed and used to build a statistical language model of full-word m-grams. In the runtime; each word in the input Arabic text is searched for in this dictionary by the "Word Analyzer and Segmentor" module. If the word is found, the word is called "analyzable" and all its existed diacritization possibilities are retrieved from the dictionary and called word analyses. A contiguous series of analyzable words in the input text is called "analyzable segment". The analyses of the words in an analyzable segment constitute a lattice, as shown in figure 3, that is disambiguated via m-grams probability estimation and A^* lattice search to infer the most likely sequence of diacritizations. The diacritized full-form words of the disambiguated analyzable segments are concatenated to the input words in the un-analyzable segments, if any, to form a less ambiguous sequence of Arabic text words. The latter sequence is then handled by the aforementioned "Factorizing Disambiguator" that is illustrated in section 3.

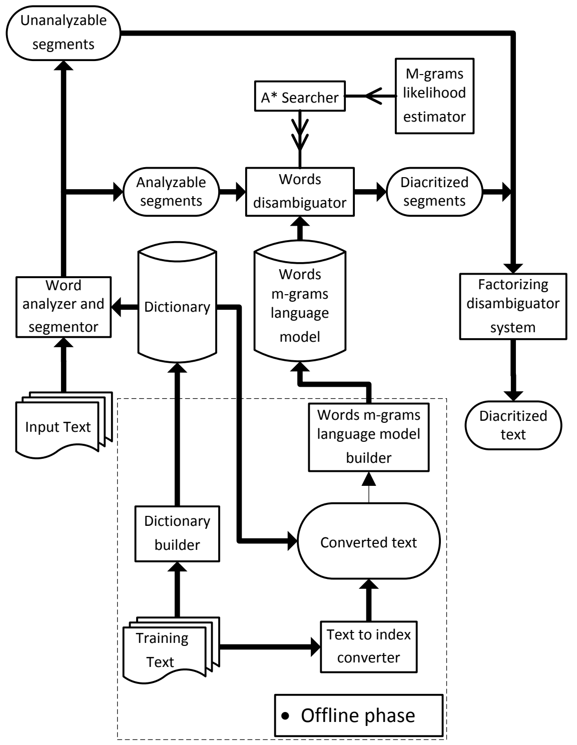

Figure 2: The hybrid Arabic diacritization architecture disambiguating factorized and full-form words.

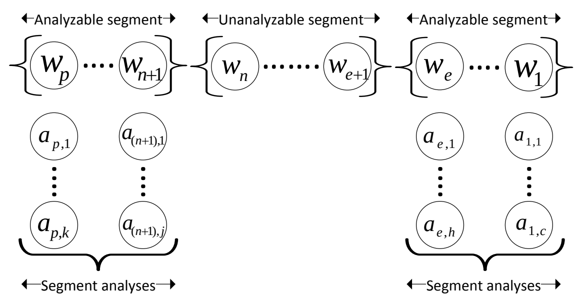

Figure 3: "Analyzable Segments" and "Un-analyzable Segments" in input text. [2]

4. Statistical Disambiguation Method

In both architectures presented on sections 2 and 3 above, the challenging ambiguity of multiple possible solutions at each word of the input text lead to the composition of a trellis abstracted in figure 4 below.

$$\begin{array}{cccc}
w_1 & w_2 & \cdots & w_L \\
\hline
a_{1,1}\bullet & a_{2,1}\bullet & \cdots & a_{L,1}\bullet \\
a_{1,2}\bullet & a_{2,2}\bullet & \cdots & a_{L,2}\bullet \\
\vdots & \vdots & & \vdots \\
a_{1,J_1}\bullet & a_{2,J_2}\bullet & \cdots & a_{L,J_L}\bullet \\
\hline
\underline{a}_1 & \underline{a}_2 & \cdots & \underline{a}_L
\end{array}$$

Figure 4: The ambiguity of multiple solutions of each word in the input text $\underline{W}$ leading to a solution trellis of possible analyses ($\underline{a}_1 \times \underline{a}_2 \times \dots \times \underline{a}_L$).

To resolve this ambiguity and infer the most statistically sound sequence of solutions; $\hat{\underline{I}}$, we rely on the well established approach of maximum a posteriori probability (MAP) estimation [2], [4], [11], [13], [15] famously formulated by:

$$\hat{\underline{I}} = \arg\max_{\forall \underline{I}} \{P(\underline{I} \mid \underline{O})\} = \arg\max_{\forall \underline{I}} \left\{ \frac{P(\underline{O} \mid \underline{I}) \cdot P(\underline{I})}{P(\underline{O})} \right\} = \arg\max_{\forall \underline{I}} \{P(\underline{O} \mid \underline{I}) \cdot P(\underline{I})\} \quad \text{…Eq (4.1)}$$

Where ($\underline{O}$) is the output observations and ($\underline{I}$) is the input observations. In other pattern recognition problems like Optical Character Recognition (OCR) and automatic speech recognition (ASR), the term *P($\underline{O}$|$\underline{I}$)* referred to as the likelihood probability, is modeled via probability distributions; e.g. HMM in ASR. Our aforementioned language factorization models and/or dictionary retrieval enable us to do better by viewing the available formal structure, in terms of probabilities, as a binary decision; i.e. a decision of whether the observation obeys the formal rules or not. This simplifies MAP formula above into:

$$\hat{\underline{I}} = \arg\max_{\forall \underline{I} \in \Re} \{P(\underline{I})\} \quad \text{…Eq (4.2)}$$

Where $\Re$ is the space of factorization model or dictionary, and *P($\underline{I}$)* is the independent probability of the input which is called the statistical language model (SLM). The term *P($\underline{I}$)* then expresses the m-grams probability estimated according to the distributions computed from the training corpus.

Using the chain rule for decomposing marginal into conditional probabilities, the term *P($\underline{I}$)* may be approximated by:

$$P(\underline{Q}) \cong \prod_{i=1}^{L} P(a_i \mid a_{i-N}^{i-1}) \quad \text{…Eq (4.3)}$$

Where *N* is maximum affordable m-gram length in the SLM and L is the number of input observations.

These conditional probabilities are simply calculated via the famous *Bayesian* formula. However, the severe *Zipfian* sparseness of m-grams of whatever natural language entities necessitates more elaboration. So, the *Good-Turing* discount and *back-off* techniques are also deployed to obtain reliable estimations of rarely or never occurring events respectively [2], [4], [11], [12], [15]. These techniques are used for both building the discrete distributions of linguistic entities from labeled corpora, and also for estimating the probabilities of any given m-gram of these entities in the runtime.

Using a variant of A^*-based algorithm; e.g. beam search, is the best known way for obtaining the most likely sequence of analyses among the exponentially increasing space $\boldsymbol{S} = \underline{a}_1 \times \underline{a}_2 \times \dots \times \underline{a}_L$ of possible sequences (paths) implied by the trellis's topology in light of the MAP formula by obtaining:

$$\hat{\underline{Q}} = \arg\max_{\text{S}} \left\{ P\left(a_{1,j_1}^{L,j_L}\right) \right\} = \arg\max_{\text{S}} \left\{ \prod_{i=1}^{L} P\left(a_{i,j_i} \mid a_{(i-h),j_{(i-h)}}^{(i-1),j_{(i-1)}}\right) \right\} = \arg\max_{\text{S}} \left\{ \sum_{i=1}^{L} \log P\left(a_{i,j_i} \mid a_{(i-h),j_{(i-h)}}^{(i-1),j_{(i-1)}}\right) \right\} \quad \text{…Eq (4.4)}$$

To obtain the sequence realizing this maximization, the A^* algorithm follows a best-first path strategy while selecting the path (through the trellis) for expanding next. This best-first strategy is interpreted in the sense of the statistical score of the path till its terminal expansion node $a_{k,j}$ given by:

$$g\left(k, a_{k,j_k}\right) = \sum_{i=1}^{k} \log P\left(a_{i,j_i} \mid a_{(i-N+1),j_{(i-N+1)}}^{(i-1),j_{(i-1)}}\right) \quad \text{…Eq (4.5)}$$

To realize maximum search efficiency; i.e. minimum number of path expansions, a heuristic function

[2] The direction of words in the text in this figure is considered from right to left.

(typically called h^*) is added to the g function while selecting the next path to expand during the A^* search so that:

$$f^*\left(k, a_{k,j_k}, L\right) = g\left(k, a_{k,j_k}\right) + h^*\left(k, a_{k,j_k}, L\right) \qquad \ldots\text{Eq (4.6)}$$

To guarantee the admissibility of A^* search; i.e. the guarantee for the search to terminate with the path with maximum score, the h^* function must not go below a minimum upper bound of the probability estimation of the remainder of the nodes sequence in the path being expanded. For our problem this function is being estimated according to:

$$h^*\left(k, q_{k,j_k}, L\right) = \begin{cases} \sum_{i=k+1}^{L} \log(P_{\max,N}) = (L-k)\cdot\log(P_{\max,N}); & L \ge N, k \ge N-1 \\ \sum_{i=N}^{L} \log(P_{\max,N}) + \sum_{i=k+1}^{N-1} \log(P_{\max,i}) & \\ \quad = (L-N+1)\cdot\log(P_{\max,N}) + \sum_{i=k+1}^{N-1} \log(P_{\max,i}); & L \ge N, k < N-1 \\ \sum_{i=k+1}^{L} \log(P_{\max,i}); & L < N \end{cases}$$

…Eq (4.7)

Where $P_{max,N}$ is the maximum probability of all input observations at the maximum affordable m-gram length in the SLM and $P_{max,i}$ is the maximum probability of all input observations at m-gram length $i < N$. Refer to [2], [4], [12], [13], [15] for proofs and full details on the statistical disambiguation methods reviewed here.

5. Related Work

Among the other recent work on the tough problem of Arabic diacritization, two groups have made remarkable attempts.

- I. Zitouni et al. (2006) [18] follow a statistical model based on the framework of maximum entropy. Their model combines different sources of information ranging from lexical, segment-based, and PoS features. They use statistical Arabic morphological analysis to segment each Arabic word into a prefix, a stem, and a suffix. Each of these morphemes is called a segment. PoS features are then generated by a parsing model that also uses maximum entropy. All these features are then integrated in the maximum entropy framework to infer the full diacritization of the input words sequence. [18]

- N. Habash and O. Rambow (2007) [9] use Morphological Analysis and Disambiguation of Arabic (MADA) system [10]. They use case, mood, and nunation as features, and use the Support Vector Machine Tool (SVMTool) [8] as a machine learning tool. They then build an open-vocabulary SLM with Kneser-Ney smoothing using the SRILM toolkit [18]. Habash & Rambow made experiments using the full-form words and a lexemes (prefix, stem, and suffix) citation form. The best results we compare with in table 5 below are the ones they obtain with the lexemes form with trigram SLM [9].

6. Experimental Results

6.1. Experimental Setup

The annotated DB used to train our aforementioned Arabic diacritizers consist of the following packages:

I- A standard Arabic text corpus with as size ≈ 750K words collected from numerous domains over diverse domains. This package is called TRN_DB_I.

It should be noted that the text of each domain is collected from several sources. This text corpus is morphologically analyzed, PoS tagged, and phonetically transcripted. All these kinds of annotations are manually revised and validated [17].

II- An extra standard Arabic text corpus with as size ≈ 2500K words that is only phonetically transcripted in full without any extra annotation. This corpus is mainly extracted from classical Islamic literature. This package is called TRN_DB_II[3]. This kind of annotation is done manually but with just one revision. So, it might contain some errors that could be a source of some errors.

III- The test data is rather challenging. It consists of 11K words that are manually annotated for morphology, PoS and phonetics. This test text covers diverse domains. This test package is called TST_DB.[4] The text of TST_DB is collected from several sources other than those used to obtain the text of TRN_DB_I and TRN_DB_II.

The three experiments discussed below have been conducted to evaluate the performance of our Arabic diacritization via both the two architectures presented in this paper; the one disambiguating factorized text features - called "Factorizing Diacritizer" - and the hybrid one – called "Hybrid Diacritizer".

6.2. Experiments Design & Results Analysis

6.2.1. Experiment no. 1:

This experiment compares the diacritization accuracy of the two architectures with both relying on SLM's built from the same Arabic text corpus. The change of diacritization accuracy of both with the gradual increase of training corpus size is also sensed. All these measure are registered in table 3 below.

---

[3] http://www.RDI-eg.com/RDI/TrainingData is where to download TRN_DB_II.

[4] http://www.RDI-eg.com/RDI/TestData is where to download TST_DB.

Training corpus size	Morphological errors		Syntactical errors	
	Factorizing diacritizer	Hybrid diacritizer	Factorizing diacritizer	Hybrid diacritizer
128k	11.5%	9.2%	26.1%	21%
256k	11.8%	7.9%	25.6%	18.7%
512k	9.9%	6.5%	23.3%	16.8%
750k	7.5%	7.0%	24.6%	16.0%

Table 3: Morphological & syntactic diacritization accuracies of the factorizing diacritizer versus the hybrid one.

These results show that the hybrid diacritizer outperforms the factorizing one with the mentioned training and test data. While the difference between the syntactical diacritization error rates is clearly wide, the difference between the morphological error rates is much closer and is vanishing with the increase of training data.

So, one may also speculate that the accuracy of the factorizing diacritizer may catch that of the hybrid one with a much more increase in the size of the training data that is needed to capture the more complicated behavior of the Arabic syntactic phenomenon than the Arabic morphological one.

Unfortunately, at the moment we do not have more annotated data of the type of TRN_DB_I that is needed to build the language models for the factorizing diacritizer.

6.2.2. Experiment no. 2:

As the training data of type TRN_DB_II is less expensive to afford than that of type TRN_DB_I, we could afford a training corpus of the former type of size 2500K words. So, the un-factorizing part of the hybrid diacritizer can rely on SLM's from up to 750K + 2500K words.

The factorizing diacritizer can of course not benefit from training data beyond that of the annotated 750K words of TRN_DB_I.

This experiment hence aims to study the effect of increasing the training data size in the un-factorizing SLM on the error rate of the hybrid Arabic diacritizer. Table 4 below shows the obtained measured error rates.

Training corpus size	Morphological errors	Syntactical errors
Size Of(TRN_DB_I) = **750K words**	7.0%	16.0%
Size Of(TRN_DB_I) + ½ Size Of(TRN_DB_II) = **2000K words**	4.9%	13.4%
Size Of(TRN_DB_I) + Size Of(TRN_DB_II) = **3250K words**	3.6%	13.0%

Table 4: Morphological and syntactic diacritization error rate of the hybrid diacritizer at large training data.

This experiment reveals that the syntactical diacritization accuracy seems to asymptote its saturation at training corpora exceeding 2000K words.

It seems that it is hard to get further significant enhancement via statistical means alone by increasing the training corpus. Achieving error rates below that 13% or so seems to need some genuine merger with more linguistically informed tools.

6.2.3. Experiment no. 3:

The architecture of the hybrid diacritizer has been explained in section 4 above where input words not found in the full-form words dictionary (also called out of vocabulary (OOV) words) are handled by the factorizing diacritizer within the statistical context of neighboring diacritized words retrieved from that dictionary. The diacritization word error rate of the hybrid diacritizer (WER_h) has hence two components; the un-factorizing one ($WER_{un\text{-}fac}$) and the factorizing one (WER_{fac}); $WER_h = WER_{fac} + WER_{un\text{-}fac}$.

As it is insightful to know the share of both $WER_{un\text{-}fac}$ and WER_{fac} in WER_h, all these rates are measured for the hybrid diacritizer running on SLM built from the largest available training data sets; i.e. TRN_DB_I + TRN_DB_2.

These measurements are given by table 5 below:

Training corpus size	Ratio of OOV	Morphological Errors		Syntactical Errors	
		WER_{fac}	WER_h	WER_{fac}	WER_h
Size Of(**TRN_DB_I**) + Size Of(**TRN_DB_II**) = **3250K words**	13.7%	2.1%	3.6%	8.1%	13.0%

Table 5: Shares of the factorizing and un-factorizing diacritization error rates in the hybrid diacritization error rate.

7. Conclusion and Future Work

It has got clear after our extensive research and experimentation on the tough problem of full Arabic diacritization that the best strategy to realize usable results is to marry statistical methods with linguistic factorization ones; e.g. morphological analysis and PoS tagging. Fully non factorizing statistical methods working on full-form words are faster to learn but suffer from poor coverage (OOV) which can be complemented by linguistic factorization analyzers.

Moreover, there seems to be an asymptotical error margin that cannot be squeezed by the state-of-the-art systems including ours esp. for syntactical diacritization without some assistance of a higher-level NLP layer(s); e.g. semantic analysis. After all, syntactic diacritization (case ending) is a projection of a hierarchical grammatical phenomenon that cannot be fully modeled via the statistical inference of linear sequences whatever long is its horizon.

Our presented system shows competent error margins with other state-of-the-art systems attacking the same problem. It has a clear plus with morphological diacritization. Moreover, when one account for the sophistication of our training and test data vs. the reported training and test data used with the other systems, some extra credit may be given to ours esp. under realistic conditions.

References

I- References in English

[1] M. Attia, M. Rashwan, A. Ragheb, M. Al-Badrashiny, H. Al-Basoumy, S. Abdou, *A Compact Arabic Lexical Semantics Language Resource Based on the Theory of Semantic Fields*, Lecture Notes on Computer Science (LNCS): Advances in Natural Language Processing, Springer - Verlag Berlin Heidelberg; www.SpringerOnline.com, LNCS/LNAI; Vol. No. 5221, Aug. 2008.

[2] M. Attia, *Theory and Implementation of a Large-Scale Arabic Phonetic Transcriptor, and Applications*, PhD thesis, Dept. of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, Sept. 2005.

[3] M. Attia, M. Rashwan, *A Large-Scale Arabic PoS Tagger Based on a Compact Arabic PoS Tags-Set, and Application on the Statistical Inference of Syntactic Diacritics of Arabic Text Words*, Proceedings of the Arabic Language Technologies and Resources Int'l Conference; NEMLAR, Cairo, 2004.

[4] M. Attia, M. Rashwan, G. Khallaaf, *On Stochastic Models, Statistical Disambiguation, and Applications on Arabic NLP Problems*, The Proceedings of the 3rd Conference on Language Engineering; CLE'2002, by the Egyptian Society of Language Engineering (ESoLE); www.ESoLE.org.

[5] M. Attia, *A Large-Scale Computational Processor of the Arabic Morphology, and Applications*, M.Sc. thesis, Dept. of Computer Engineering, Faculty of Engineering, Cairo University, 2000.

[6] V. Cavalli-Sforza, A. Soudi, T. Mitamura, *Arabic Morphology Generation Using a Concatenative Strategy*, ACM International Conference Proceeding Series; Proceedings of the first conference on North American chapter of the Association for Computational Linguistics (ACL), 2000.

[7] J. Dichy, M. Hassoun, the *DINAR.1 (DIctionnaire INformatisé de l'ARabe, version 1) Arabic Lexical Recourse, an outline of contents and methodology*, The ELRA news letter, April-June 2005, Vol.10 n.2, France.

[8] J. Giménez, L. Màrquez, *SVMTool: A General PoS Tagging Generator Based on Support Vector Machines*; The proceedings of LREC'04, 2004.

[9] N. Habash, O. Rambow, *Arabic Diacritization through Full Morphological Tagging*, Proceedings of the 8th Meeting of the North American Chapter of the Association for Computational Linguistics (ACL); Human Language Technologies Conference (HLT-NAACL) 2007.

[10] N. Habash and O. Rambow, Arabic Tokenization, *Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop*, Proceedings of ACL'05, 2005.

[11] D. Jurafsky, J. H. Martin, *Speech and Language Processing; an Introduction to Natural Language Processing, Computational Linguistics, and Speech Processing*, Prentice Hall, 2000.

[12] S. M. Katz, *Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer*, IEEE Transactions on Acoustics, Speech and Signal Processing, vol. ASSP-35 no. 3, March 1987.

[13] N. J. Nilsson, *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill, 1971.

[14] A. Ratenaparkhi, *Maximum Entropy Models for Natural Language Ambiguity Resolutions*, PhD thesis in Computer and Information Science, Pennsylvania University, 1998.

[15] H. Schütze, C. D. Manning, *Foundations of Statistical Natural Language Processing*, the MIT Press, 2000.

[16] A. Stolcke, *(SRILM) An Extensible Language Modeling Toolkit*, The Proceedings of the International Conference on Spoken Language Processing (ICSLP), 2002.

[17] M. Yaseen, et al., *Building Annotated Written and Spoken Arabic LR's in NEMLAR Project*, LREC2006 conference http://www.lrec-conf.org/lrec2006, Genoa-Italy, May 2006.

[18] I. Zitouni, J. S. Sorensen & R. Sarikaya, *Maximum Entropy Based Restoration of Arabic Diacritics*, Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (ACL); Workshop on Computational Approaches to Semitic Languages; Sydney-Australia, July 2006; http://www.ACLweb.org/anthology/P/P06/P06-1073.

II- References in Arabic

[19] (A. Arragehy, 1993) التَّطبيقُ الصَّرْفيُّ، عَبْدُهُ الرَّاجِحيّ، دارُ المَعْرفَةِ الجَامِعِيَّةِ، الإسْكَنْدَرِيَّةُ، 1993