AUTOMATIC ARABIC TEXT DIACRITIZATION USING RECURRENT NEURAL NETWORKS

By
Alaa Khaled Radwan Arabiyat

Supervisor
Dr. Gheith Ali Abandah

This Thesis was Submitted in Partial Fulfillment of the Requirements for the Master's Degree of Science in Computer Engineering and Networks

Faculty of Graduate Studies
The University of Jordan

May, 2015

COMMITTEE DECISION

This Thesis (Automatic Arabic Text Diacritization Using Recurrent Neural Networks) was successfully defended and approved on 4/5/2015

Examination Committee	**Signature**
Dr. Gheith Abandah, (Supervisor) Assoc. Prof. of Computer Architecture	----------
Dr. Iyad Jafar, (Member) Assoc. Prof. of Digital Image Processing	----------
Dr. Omar Al-Kadi (Member) Assoc. Prof. of Pattern Recognition	----------
Dr. Ali Al-Haj (Member) Assoc. Prof. of Digital Image Processing (Princess Sumaya University for Technology)	----------

DEDICATION

To my parents for their prayers and encouragements

To my children Hala, Hamzeh, and my six month little boy Shaher

To my beloved husband Rami

ACKNOWLEDGEMENT

I would like to express my significant gratitude and great thanks to my advisor Dr. Gheith Abandah for his generous efforts and patience in supervising this thesis. He is an excellent mentor and he taught me how to really do research. I am grateful for having been his student.

Many thanks are also due to all staff in the department of Computer Engineering in the University of Jordan for their cooperation, help and understanding throughout my Master studies.

List of Contents

List of Figures

List of Tables

List of Abbreviations

ASR	Automatic Speech Recognition
ATB3	LDC Arabic Treebank, Part 3, version 1.0
BPTT	Back-Propagation Through Time
BAMA	Buckwalter Arabic Morphological Analyzer
BRNN	Bidirectional Recurrent Neural Network
DNN	Deep neural networks
DER	Diacritic Error Rate
HMM	Hidden Markov Model
LDC	Linguistic Data Consortium
MLPs	Multilayer Perceptrons
MSA	Modern Standard Arabic
NLP	Natural Language Processing
POS	Part of Speech
RNN	Recurrent Neural Network
SHR	Simple Heuristic Rules
SVM	Support Vector Machine
TTS	Text-To-Speech
TDNNs	Time Delay Neural Networks
WER	Word Error Rate

AUTOMATIC ARABIC TEXT DIACRITIZATION USING RECURRENT NEURAL NETWORKS

By
Alaa Khaled Radwan Arabiyat

Supervisor
Dr. Gheith Ali Abandah

ABSTRACT

Nowadays, Arabic documents are often found undiacritized in schools, universities, workplaces, books, and the media. This style is common to cut the typing costs. As a consequence, many words become ambiguous because they could have many diacritization variants with same word consonants. Native speakers can generally infer the correct pronunciation and the intended meaning of a word from their intuitive knowledge of the language and from the context. But non-native speakers, children, and Arabic software applications such as Automatic Speech Recognition (ASR), and Text-to-speech (TTS) systems need full diacritized texts.

This problem has been tackled by many researchers who tried to restore the missing diacritics automatically using rule-based, statistical, and hybrid approaches. Each of the first two approaches has its own advantages that when combined together could give better performance.

In this thesis, we are investigating a novel approach of utilizing recurrent neural networks (RNN) to restore diacritics using a sequence transcription network of deep bidirectional LSTM (long short-term memory). This statistical approach has reduced the error rates over the best published results. We have also implemented three other hybrid schemes where in each one; we add linguistic information to the input sequences to help RNN transcription learning. Several correction techniques are also applied to the RNN results which contributed in enhancing the diacritization accuracy rates even better.

The linguistic supplement to the RNN leads to a state-of-the-art hybrid diacritization model. Using LDC's Arabic Treebank Part 3 corpus, we achieve a diacritic error rate of 2.74%, and a word error rate of 9.66%. When ignoring the diacritization error in the last letter of each word, we obtain a diacritic error rate of 1.24%, and a word error rate of 3.95%.

CHAPTER I

Introduction

In this preliminary chapter; the importance of automatic text diacritization is discussed along with potential applications, a necessary demonstration of diacritization types is presented next, research objectives and contributions are then manifested, and the rest of the thesis is finally outlined.

1.1 Importance of Arabic Text Diacritization

The modern written language (Modern Standard Arabic, MSA) is derived from the Classical Arabic (CA) with new words added to it continually to meet the present and future needs of the language to express the evolution in science and society (Farghaly and Shaalan,2009).

MSA texts are normally written without diacritics. Native speakers can infer the correct pronunciation of the script from the context and the speaker's knowledge of the grammar and lexicon of the language. However, the lack of diacritics causes ambiguity for non-native speakers and children in their educational beginnings.

Diacritic marks are used to add phonetic information to the alphabet letters. These marks could be written above or below the letter. Diacritics can be divides into three categories: short vowels, nunation, and syllabification marks (Azmi and Almajed, 2013). Table 1 goes over the basic diacritization marks. The three short vowels could be placed on any constant of the word. Nunation diacritics or sometimes named double case ending diacritics are only placed on last letter of the word. The Shaddah, or germination diacritic, is pronounced as consonant doubling and could be combined with any other diacritics. Sukon is used to show that the letter does not contain vowels. The

Buckwalter transliteration of diacritics is also mentioned because we are going to use Buckwalter morphological analyzer in this work and it has special transliteration for Arabic letters and diacritics.

Table 1. The basic Arabic diacritics

Diacritic category	Diacritic name	Shape	Pronunciation	Buckwalter Transliteration
Short vowels	Fatha	◌َ	/a/	a
	Damma	◌ُ	/u/	u
	Kasra	◌ِ	/i/	i
Nunation	Tanween fath	◌ً	/an/	F
	Tanween damm	◌ٌ	/un/	N
	Tanween kasr	◌ٍ	/in/	K
Syllabification marks	shaddah	◌ّ	consonant doubling	~
	sukon	◌ْ	vowel absence	o

Language software applications benefit a lot from a diacritized text to correctly process data, Automatic Speech Recognition (ASR), and Text-to-speech (TTS) systems are few examples. These systems are the bases of many industrial applications like Interactive Voice Response (IVR) systems and screen readers for the blind and many other NLP applications (Anastasiou, 2011). Moreover, when searching for an Arabic word, many unrelated words would appear in the search results because of the lack of diacritization. Also, machine translation from and to Arabic need a diacritized text to get the correct translation.

The lack of diacritical marks not only add ambiguity to the Arabic text but it also make the same effect on all other languages that uses the Arabic alphabet including Persian, Kurdish,urdu, and jawi (Lewis, 2009).

Many people criticize the Arabic writing system and accuse it with being difficult to learn with. They claim that in other languages “you read to understand” but in Arabic “you need to understand in order to read”. Also they asked for Arabic writing that have complete match of Arabic speech. The lack of diacritization is the cause of

these problems (Abu-Hamedeh, 2009). This is a serious problem especially if you know that most MSA texts are not diacritized mainly to increase the typing speed. In addition to this, placing diacritics manually is inefficient and time consuming method.

For all the reasons mentioned above, we realize that computer processing of Arabic text depends heavily on diacritized text and is no longer marginal or minor thing.

1.2 Diacritization Types: Lexemic and Inflectional

The problem with the undiacritized Arabic script is that the written word may not be correctly pronounced depending on the orthographic representation only. The reason behind this is the language's highly inflective and derivative nature where many words can be generated from the same combination of consonants with maybe different meanings and different pronunciations. Diacritics are needed to indicate the intended pronunciation.

The diacritics can be classified into two kinds (Habash and Rambow, 2007): lexemic diacritics (morphology-dependent), and inflectional diacritics (syntax-dependent).

Lexemic diacritics distinguish between lexemes that have the same spelling. For example, the word (**علم**) has six possible morphological dictionary forms of diacritizations each of them has different meaning. Table 2 reviews the six forms that were generated using Buckwalter Morphological Analyzer BAMA (Buckwalter, 2004). The words pronunciations are written using Buckwalter transliteration (see Appendix A). The diacritics generated here are considered semantic-dependent. POS tags are used to distinguish between the different solutions. The first solution: عَلِمَ Ealima is tagged as 3rd Person Masculine, Singular, Perfective Verb; عُلِمَ Eulima for 3rd Person Singular, Passive Verb, and عَلَّمَ Eal~ama for the Intensifying, Causative, Denominative Verb. The

three last solutions are tagged as nouns and they differ by meaning.

Table 2. Possible diacritized dictionary forms of **علم**

No.	Script	Pronunciation(BW)	meaning
1	عَلِمَ	Ealima	he know
2	عُلِمَ	Eulima	be known
3	عَلَّمَ	Eal~ama	teach
4	عِلْم	Eilom	knowledge/knowing
5	عِلْم	Eilom	science/study of
6	عَلَم	Ealam	flag

On the other hand, the same word may have different syntactic rules depending on its role in the parsing tree of its sentence, and is typically expressed on the final consonant of a word. The diacritics generated here are called inflectional diacritics and are considered syntax-dependent. It varies according to the nominal case or verbal mood. A single wrongly predicted case ending has the capacity to completely reverse the intended meaning. Ayah 28 from Fatir chapter in the holly Qur`an in Figure 1 is a clear example on the importance of correct diacritization. The first sentence implies the syntactic diacritic of the target word اللَّهَ -which is an "object" in the parsing tree – is "Fatha", while the second one implies the syntactic diacritic of the target word اللهُ– which is a "subject" in the parsing tree – is "Damma". The wrong diacritics on the last consonant of the words (الله) and (العلماء) completely reversed the meaning.

Only those fear Allah, from among His servants, who have knowledge.	إِنَّمَا يَخْشَى اللَّهَ مِنْ عِبَادِهِ الْعُلَمَاءُ
Allah fears only, from among His servants, those who have knowledge.	إِنَّمَا يَخْشَى اللَّهُ مِنْ عِبَادِهِ الْعُلَمَاءَ

Figure 1. An example on the significance of inflectional diacritization

1.3 Research Objectives and Contributions

As we mentioned before, the human brain of native Arabic speakers can infer the correct pronunciation and the intended meaning of the script. It has been found that the human brain is a Recurrent Neural Network (RNN) consisting of neurons with feedback connections that can learn many behaviors. This was the motivation for us to try testing a novel approach of using RNN to automatically diacritize Arabic text (Pineda, 1987).

We can assemble the human brain with artificial RNNs using general computers which can learn sequence processing tasks to map input sequences to output sequences. The network is trained through the interaction of previous experiences where we give the input, the undircritized Arabic text and the expected target of this input, which is the fully dicritized Arabic text, to train the RNN under supervised learning. Once this is done we can give the RNN any undircritized input text and get the fully diacritized output text.

This method is very powerful compared to other adaptive approaches such as Hidden Markov Models HMM (no continuous internal states), feedforward networks FFN or Support Vector Machines SVM (no internal states at all).

RNN has been used with many sequence problems such as handwriting recognition (Abandah, et al., 2014) and speech recognition (Graves, et al., 2013) and achieved fabulous results.

Our novel approach is to use RNN in automatic Arabic text diacritization. We have tested this approach on a previous work (Abandah, et al., 2015) and achieved a state of art results in the field. The main contribution of this thesis is to improve these results even more by developing a hybrid system that combines the morphology based

diacritizer with the statistical RNN diacritizer.

The objectives and contributions of this project are summarized as follows:

1. Investigating the use of recurrent neural network (RNN) in automatic diacritization of the Arabic texts (Abandah, et al., 2015).

2. Building an accurate system to restore all Arabic diacritics (the three short vowels: fatha, kasra, and damma, the three tanweens, the shadda, and the sukoon) and comparing it with the best performing reported systems results using similar training and validation corpus for the sake of fair comparison.

3. Improving the RNN diacritization accuracy by supplementing the RNN input with linguistic information as a pre-processing step and applying correction techniques to the RNN output as a post-processing step.

Finally, we believe that this thesis is a useful addition to the field of Arabic text diacritization and a step towards the comprehensive computer support of the Arabic language.

1.4 Thesis Outline

After this introductory chapter; Chapter 2 reviews previous related work and past approaches used to automatically diacritize Arabic text including rule-based, statistical, and hybrid approaches.

Chapter 3 describes the technologies used in our system including RNN that has been used in many sequence transcription tasks and recently used it in automatic diacritization. BAMA tool is also described which is widely used in previous works.

In Chapter 4 the methodology is described including the datasets, data pre-processing, sequence transcription, RNN training parameters, and the data post-processing.

We give detailed descriptions for the conducted experiments of the four implemented systems in Chapter 5. Also results comparison with recent related work is made.

Finally, the last chapter presents overall conclusions, observations, and suggestions for future work.

CHAPTER II

Literature Review

Restoration of diacritics is an active area in the current Natural Language Processing (NLP) literature. Most NLP tasks benefits a lot when using fully diacritized texts. Many researchers have tried several methods to tackle this problem. However, there is still a room to improve the accuracy of adding diacritics. Past techniques to automatically diacritize Arabic text can be divided mainly into three categories: rule-based, statistical, and hybrid approaches (Azmi and Almajed, 2013). The following sections give a brief description of the approaches that were state-of-art at the times when they were published in.

2.1 Rule-based Approaches

Automatic diacritization was initially solved using rule-based approaches. These approaches used morphological analyzers, dictionaries, and grammar modules. El-Sadany and Hashish (1988) system uses a dictionary, analyzer, and a grammar module that contains morphophonemic and morphographemic rules. Another, rule-based method was presented by El-Imam (2004) as an essential intermediate step in the process of letter to sound mapping. Shaalan (2010) also developed a rule-based morphological and syntax analyzers. These analyzers depend on the linguistic knowledge only and could be used to predict the missing diacritics.

Rule-based approaches are complicated and need efforts to build morphological, syntax, and semantics analyzers. Also, their limited ability to maintain up-to-date rules and extending them to other Arabic dialects made them insufficient for diacritization.

Also, new words are always generated in living languages; therefore using rule-based methods does not sustain for long period.

2.2 Statistical Approaches

With the beginning of the new century, there have been several studies that use statistical methods to solve the diacritization problem. These methods do not require any specific knowledge of the language, so they can be applied on Arabic text as well as any other language that use diacritical marks.

Gal in (Gal, 2002) used the Hidden Markov Models (HMMs) to solve the problem. The HMM approach is a statistical approach used to capture the contextual correlation among words. It consists of hidden states that represent diacritized words from the training corpus, and produce undiacritized word observation. Then Viterbi algorithm works on these observation sequences to predict diacritics. Gal`s approach restores only short vowels (a subset of all diacritics). Gal achieved a word accuracy of 86%, where most of the errors were due to words that were not found in the training corpus.

Kirchhoff et al. (2002) targeted improving Arabic speech recognition and investigated the use of diacritization to achieve their goal. They were also interested in improving dialectal Arabic recognition in addition to formal Arabic recognition. They used the LDC CallHome ECA dialectal corpus, which was distributed with both diacritized and undiacritized transcriptions, to derive diacritics using maximum-likelihood unigram prediction. The results showed that the MSA word error rate is 28% to 9%, depending on whether or not case ending diacritics are counted.

Hifny (2012) used statistical trigram language model, smoothing techniques, and dynamic programming (DP) to restore diacritics. Each potential diacritized word sequence of an undiacritized input sentence carries a probability value set by the n-gram models. Dynamic programming algorithm is used to select the most probable sequence. Several smoothing techniques were used to handle the problem of unseen n-grams in the training data. Hifny using Tashkeela corpus (Zerrouki 2011) achieved 3.4 and 8.9% for diacritic and word error rates, respectively.

Azim et al. (2012) diacritization system requires the availability of speech input as it combines acoustic information from the speech input model based on HMM to the text-based model based on Conditional Random Fields. The diacritic and word error rates on the Linguistic Data Consortium's Part 3 of the Arabic Treebank of diacritized news stories (LDC ATB3) (Maamouri et al., 2004) are 1.6 and 5.2%, respectively. Although these are great results; this work is not very practical since you need the speech input which is not available for every text you want to diacritize.

To the best of our knowledge, the most accurate statistical approach is due to (Abandah, et al., 2015). This approach is based on the deep bidirectional long short term memory architecture. It uses RNN sequence transcription to automatically add diacritics to Arabic text. They achieved 2.72 and 9.07% diacritic and word error rates respectively, on LDC ATB3. Using Tashkeela corpus the results were even better since the size of the corpus is much bigger and achieved 2.09 and 5.82% for diacritic and word error rates respectively.

2.3 Hybrid approaches

In (Vergyri and Kirchhoff, 2004), several knowledge sources (acoustic, morphological, and contextual) were used to automatically diacritize Arabic texts and

the effect of their combination was investigated. Diacritization is handeled as an unsupervised tagging problem where each word is tagged as one of the many possible forms provided by BAMA (Buckwalter, 2002). The Expectation Maximization (EM) algorithm is used to learn the tag sequences. They also investigated the use of Arabic dialectal speech in addition to MSA. For this study, they used two different corpora, the FBIS corpus of MSA speech and the LDC CallHome ECA corpus. Kirchoff and Vergyri did not model the shadda diacritic. They achieved a word error rate and diacritic error rate 27.3% and 11.5% respectively.

A weighted finite state machine based algorithm was proposed to restore the missing diacritics by Nelken and Shieber (2005). Their basic module consists of three language models which are used in a serious of finite state transducers to produce the most probable diacritized word when given the undiacritized one. The problem with this approach is the independence of the transducers from each other where the later transducer for instance cannot refer to any former one to get some information from it. This system was trained and tested on LDC's Arabic Treebank of diacritized news stories (Part 2) and generated a word error rate of 23.61% and a diacritic error rate of 12.79% when case endings were included. Without case endings, the results were 7.33% and 6.35% respectively.

Zitouni et al. (2006) follow a statistical model based on the framework of maximum entropy where several sources of information are used including lexical, segment-based, and Part Of Speech (POS) features. They used statistical Arabic morphological analysis to segment each Arabic word into a prefix, a stem, and a suffix using WFST approach. 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 combined in the maximum entropy framework to predict the full diacritization

of the input words' sequence. Their system trained and evaluated on LDC ATB3 and performed 18% word error rate and 5.5% diacritic error rate with case endings. Without case endings, the results were 7.9% and 2.5% respectively.

Habash and Rambow (2007) extended the use of their Morphological Analysis and Disambiguation of Arabic (MADA) system. They use the case, mood, and nunation as features because of their importance, and use the Support Vector Machine Tool (SVM Tool) as a machine learning tool. The system also used LDC ATB3. They achieved a word error rate of 14.9% and diacritic error rate of 4.8% with the case ending diacritic. Without case endings, the word error rate and diacritic error rate were 5.5% and 2.2% respectively.

The stochastic Arabic diacritizer (Rashwan and Al-Badrashiny, 2011) introduced a two-layer stochastic system to diacritize raw Arabic text automatically. The first layer predicts the most likely diacritics by choosing the sequence of unfactorized full-form Arabic word diacritizations with maximum marginal probability via A* lattice search algorithm and n-gram probability estimation. When full-form words not found, the system uses the second layer, which factorizes each Arabic word into its possible morphological components (prefix, root, pattern and suffix), then uses n-gram probability estimation and A* lattice search algorithm to select among the possible factorizations to get the most likely diacritization sequence. While the second layer has better coverage of possible Arabic forms, the first layer is faster to learn and yields better disambiguation results especially for predicting case endings diacritics. Their hybrid system exploits the advantages of both layers. The system used the same training and test corpus used by Zitouni et al. (2006), and achieved 12.5% word error rate and 3.8% diacritic error rate with case endings, and 3.1% word error rate and 1.2% diacritic error rate without case endings.

The hybrid system by Said et al. (2013) also used LDC ATB3 and achieved excellent results. This approach involve using automatic correction, morphological analysis, POS tagging, and out of vocabulary diacritization and produces diacritic and word error rates of 3.6 and 11.4%, respectively. This system uses HMMs for morphological analyses disambiguation and resolving the syntactic ambiguity to restore the syntactic diacritic.

In this thesis, we combine the use of linguistic module along with Abandah et al. (2015) statistical method. The results are very promising and they are reported in Chapter 5.

In addition to the Arabic diacritization systems implemented by academic researchers and discussed above, there are systems implemented by commercial organizations to satisfy the market requirements. Sakhr's, Xerox's, and RDI's systems are examples (Rashwan and Al-Badrashiny, 2011). These systems suffer from few drawbacks that prevent them from being widely adopted such as being based on the standard Arabic dictionaries (i.e., if the word is not registered in these dictionaries it is not considered). Also, some of these systems do not account for the correlation relationship between the word and its neighbors. Even if the two previous shortcomings are considered as in the RDI system, the long needed time for training and validation of the corpora make it ineffective.

CHAP288TER III

Technologies Used

3.1 Recurrent Neural Networks (RNN)

Neural networks in the human brain works in a very accurate and complex way. It was found that the information is not stored in the human brain in a specific location, but spread over many neurons. When one tries to remember something, the brain collects this information from all these neurons. Scientists emulated the human brain using artificial neural networks through computer software which can solve many problems especially the ones that involve learning and decision making.

Natural languages processing is one of the most difficult topics in machine learning. Nevertheless, it is progressing slowly and steadily. The use of neural networks is not very common in this field. However, many researchers achieved promising results when using them. Khedher (1999) has a beautiful statement on this: “The main reason why it seems that the Arabic text processing seems to be suitable for neural network application is that people from their early age are trained to talk properly. Why neural networks cannot be trained similarly? Of course, proper and enough data is necessary.”

The learning algorithm of a neural network can either be supervised or unsupervised. Our work uses supervised learning. The mission with this kind of learning is to infer a suitable function from labeled training data. In this work, the training data is part of the LDC ATB3 corpus that is composed of a set of training examples. Each example is a pair consisting of an input sequence (the undiacritized

sentence) and a desired target sequence (the same sentence but with diacritics). A supervised learning algorithm uses the training data and infers a function, which can be applied to new examples. These new test examples could be unseen before by the function that was found based on the training data. This requires the learning algorithm to generalize from the training data set to test data set in a reasonable way. That is why we use an extra validation set which is extracted from the training set to validate the performance of the learning algorithm during training. Actually validation sets are used to determine when training should stop, in order to prevent overfitting.

This work presents a sequence transcription approach for the automatic diacritization of Arabic text using recurrent neural networks. RNNs are used to solve the diacritization problem because they benefit from the context of the input text sequences (El Hihi and Bengio, 1995).

3.1.1 Feedforward neural network vs. Recurrent neural networks

The feedforward neural network is a simple type of artificial neural network. In this network, data passes forward in one direction from the input layer, which consists of one neuron per feature, through the hidden layers (if any), which have no cycles or loops to the output layer, which has one neuron per class as shown in Figure 2. The user determines the number of neurons and topology architecture within the hidden layer. The weights of the neural networks are adjusted frequently using algorithms such as the backpropagation that works by running the training data through the neural networks, and calculating the difference between the desired and actual outputs. The output layer then propagates these differences back to the input layer to adjust the weights of the network.

For the standard feed forward network, the output vector sequence *y = (y1, . . . , yT)* when given the input sequence *x = (x1, . . . , xT)* is computed by the following equations:

$$h_t = \mathcal{H}\,(W_{ih}x_t + b_h) \quad (1)$$

$$y_t = (W_{ho}h_t + bo) \quad (2)$$

where the *W* terms denote weight matrices (e.g., W_{ih} is the input-hidden weight matrix), the *b* terms denote bias vectors (e.g., b_h is hidden bias vector), and *H* is the hidden layer activation function (usually an element wise application of a sigmoid function) (Graves, et al., 2013).

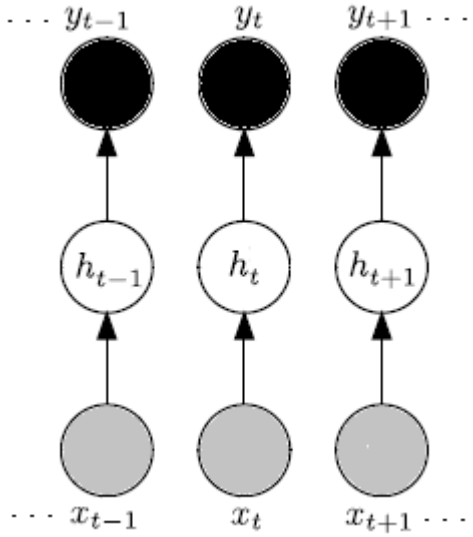

Figure 2. A simple feed forward network

The Recurrent neural network (RNN) is a different from feedforward architectures in the sense that they not only operate on the current inputs but also on the previous contents of the hidden layer for each time step. Therefore, in training, the gradient of an error function is calculated using all inputs, not the recent inputs only. This approach is known as the Back Propagation Through Time (BPTT); where the

hidden layer is updated using the external input and the activation from the previous forward propagation through an additional recurrent weight layer as shown in Figure 3.

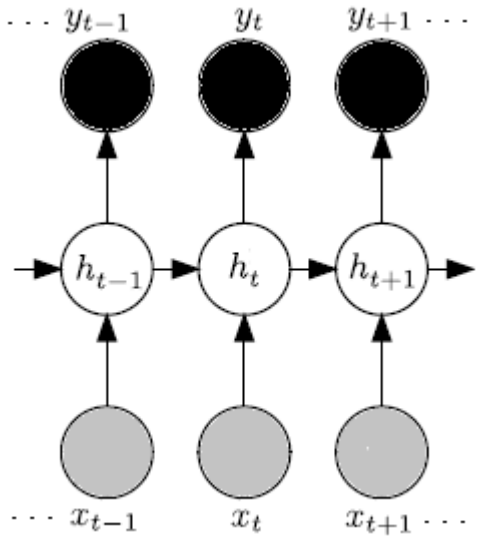

Figure 3. A simple recurrent network

This simple recurrent network is also known as Elman network (Elman, 1990). The output layer, *y*, is computed by iterating the following equations (Abandah, et al., 2015):

$$h_t = \mathcal{H}\left(W_{ih}\, x_t + W_{hh}\, h_{t-1} + b_h\right) \tag{3}$$

$$y_t = W_{ho} h_t + b_o \tag{4}$$

3.1.2 Long-Short Term Memory Network (LSTM)

In Hochreiter and Schmidhuber (1997), the long short-term memory network is proposed to address the vanishing gradient issue when using recurrent networks. This problem has been tackled by replacing a subset of neurons (or all of them) in the network by memory cells. Figure 4 illustrates such a cell.

Each memory cell is designed with self-connections storing the temporal state of the network in addition to special multiplicative units called gates to control the flow of

information. Input gate, forget gate and output gate are multiplicative gate units that allow the cells to store and retrieve information over time, giving them access to long-range context. The linear activation and self connection of value 1 means that the gradient through this connection does not loose norm, and therefore does not vanish (Graves and Schmidhuber, 2009).

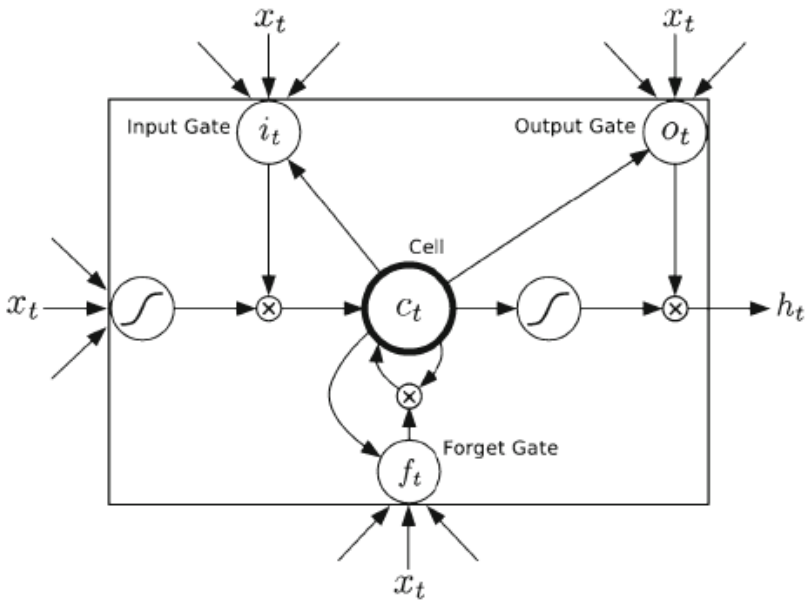

Figure 4. Long short-term memory cell

For the version of LSTM used in this research [Gers, et al., 2003), the hidden layer activation function is implemented by the following composite function :

$$i_t = \sigma\left(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i\right) \quad (5)$$

$$f_t = \sigma\left(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f\right) \quad (6)$$

$$c_t = f_t c_{t-1} + i_t \tanh\left(W_{xc}x_t + W_{hc}h_{t-1} + b_c\right) \quad (7)$$

$$o_t = \sigma\left(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o\right) \quad (8)$$

$$h_t = o_t \tanh(c_t) \quad (9)$$

where σ is the logistic sigmoid function and i , f , c, and o, respectively, are the input gate, forget gate, cell activation, and output gate vectors, all of which are of the same size as the hidden vector h. The weight matrix subscripts have an obvious meaning, for example, W_{hi} is the hidden-input gate matrix, W_{xo} is the input–output gate matrix. The weight matrices from the cell to gate vectors (e.g., W_{ci}) are diagonal, so element m in each gate vector only receives input from element m of the cell vector. The bias terms (which are added to i , f , c, and o) have been omitted for clarity (Abandah, et al., 2015).

3.1.3 Bidirectional Recurrent Neural Networks

RNN's proved that they can deal efficiently with sequential data that has relations between data points that are close in the sequence. In general RNN architecture, the input vectors are fed one at a time into the RNN. Instead of using a fixed number of input vectors as done in the Multilayer perceptrons (MLP's) and time delay neural networks (TDNN's) structures, this architecture can make use of all the available input data up to the current time t_c (i.e. $Wt, t=1,2,\ldots, tc$) to predict Y_{tc} .

Future input information coming up after tc is sometimes also useful for prediction. This can be done using RNN by delaying the output by a certain units of G time frames to include future information up to $Wtc+G$ to predict Ytc . Theoretically, G could be made very large to capture all the available future information, but in practice, it is found that prediction results drop if G is too big (Robinson, 1994).

To overcome this limitation of a common RNN, Schuster and Paliwal (1997) proposed a bidirectional recurrent neural network (BRNN) that can be trained using all available input information in the past and future of a specific time period. This is achieved by splitting the state neurons of a regular RNN in two parts; the first is the

forward states which is responsible for the positive time direction from t=1 to T. and the second are the backward states which is responsible for the negative time direction from t=T to 1. Outputs from forward states are not connected to inputs of backward states. The data in both directions of the BRNN are then fed forward to the same output layer. This leads to the general structure that can be seen in Figure 5.

BRNN hidden layers in the forward direction is the same as for a regular RNN, with the variation that is the input sequence is fed in opposite directions to the two hidden layers, and the output layer is not modified until the hidden layers of both directions have processed the whole input sequence. In a similar way, the backward direction of the BRNN is trained with back-propagation through time (BPTT), except that the entire output layer neurons are modified then fed back to the two hidden layers.

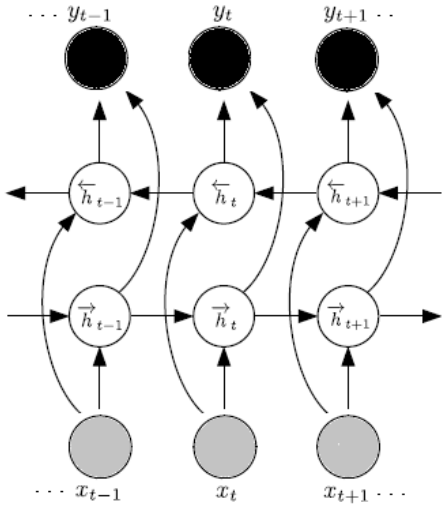

Figure 5. General structure of the bidirectional RNN

The output sequence *y* of BRNN is computed by iterating the backward layer from *t=T* to *1*, the forward layer from *t=1* to *T*, and then updating the output layer:

$$\overrightarrow{h_t} = \mathcal{H}(W_{x\vec{h}}\, x_t + W_{\vec{h}\,\vec{h}}\overrightarrow{h_{t-1}} + b_{\vec{h}}) \tag{10}$$

$$\overleftarrow{h_t} = \mathcal{H}\left(W_{x\overleftarrow{h}}\, x_t + W_{\overleftarrow{h}\overleftarrow{h}}\overleftarrow{h_{t+1}} + b_{\overleftarrow{h}}\right) \tag{11}$$

$$y_t = W_{\vec{h}y}\overrightarrow{h_t} + W_{\overleftarrow{h}y}\overleftarrow{h_t} + b_o \quad (12)$$

where $\overrightarrow{h_t}$ is the forward hidden sequence , and $\overleftarrow{h_t}$ is the backward hidden sequence (Abandah, et al., 2015) (Graves, et al., 2013).

3.1.4 Deep recurrent neural network

In the last few years, neural networks have witnessed the increase in using networks composed of multiple hidden layers, which are often described as deep neural networks (DNN). DNNs have offered powerful solution for sequence problems such as speech recognition (Graves, et al., 2013) and handwritten digit recognition (Ciresan, et al., 2010). Their success is commonly referred to the hierarchy that is introduced due to the several layers. Each layer is responsible for part of the problem, and the output of one layer is passed as an input to the next layer. Take into your consideration that features on the higher layers are more complex and are constructed from the ones on the layers below as shown in Figure 6. This process continues until reaching the final layer which formulates the final output (Hermans and Schrauwen, 2013).

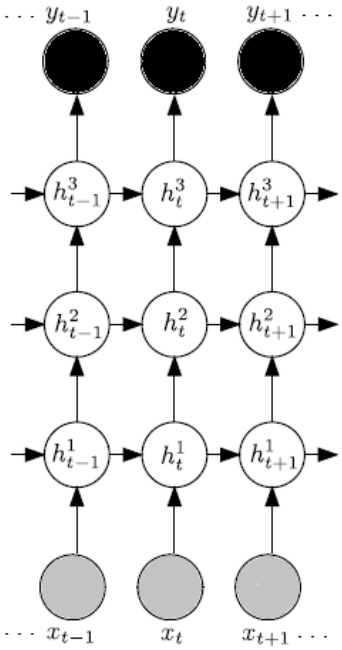

Figure 6. Deep RNN

As mentioned before, DNNs can model complex non-linear relationships. This is the solution for the common RNN problem where information passes through one layer to output the final results. The extra layers of DRNN allow composition of input features from lower layers, giving the possibility of modeling complex data with fewer units than a standard network. Deep neural architectures have achieved state-of-the-art results in many tasks in natural language processing, such as information retrieval (Huang, et al., 2013), machine translation (Cho, et al., 2014), and other NLP areas.

Assuming that the same hidden layer function is used for all *N* layers in the stack, the hidden vector sequences h^n are iteratively computed from *n=1* to *N* and *t=1* to *T*:

$$h_t^n = \mathcal{H}\left(W_{h^{n-1}h^n} h_t^{n-1} + W_{h^n h^n} h_{t-1}^n + b_h^n\right) \quad (13)$$

where $h^o = x$. The network outputs y_t are

$$y_t = W_{h^N y} h_t^N + b_o \quad (14)$$

If we use a bidirectional RNN to implement each hidden layer with LSTM nodes, then we are implementing deep bidirectional LSTM, the main architecture used in this research (Abandah, et al., 2015). Figure 7 illustrate this architecture.

Our experiments were carried out with the open source software library RNNLIB which is a recurrent neural network library for sequence labeling problems developed by Alex Graves (Graves, 2008). We experimented using the “one-to-one” network that use the “one-to-one” letter encoding described in Section 4.3.1 in which the target sequences had a one-to-one correspondence with the inputs sequences. Thus, the lengths of sequences x and y are equal as implied by equations 1 and 2 above.

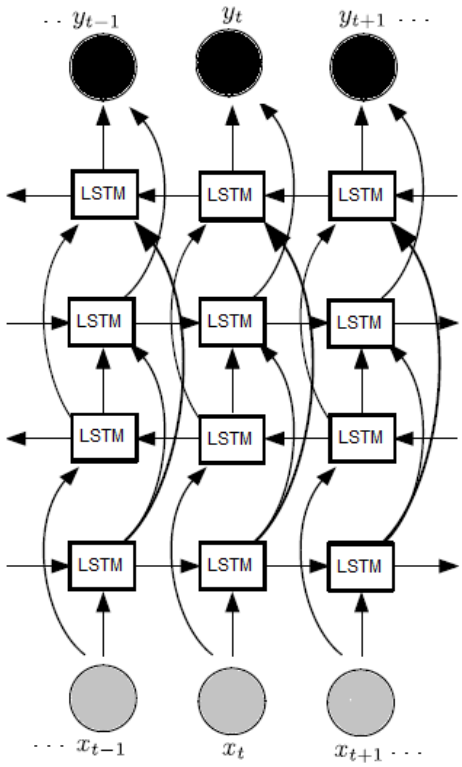

Figure 7. Deep bidirectional LSTM

The network was trained to individually classify each input letter with the corresponding diacritized version. As is standard for classification tasks, a softmax output layer was used to define a probability distribution over the output labels, and the network was trained to minimize the cross-entropy of this distribution with the target labels. That is, given a length *T* input, target sequence pair *(x, y*)*, the network outputs at time *t* are interpreted as the probability *Pr(k|t, x)* of emitting (diacritized) letter *k* at time *t* and the loss function minimized by the network is defined as *L(x, y*)* = - $\sum_{t=1}^{T} \log \Pr(y_t^*|t,x)$. The network is trained to minimize the loss function *L* using online gradient descent algorithm with momentum. A similar approach was previously used for bidirectional LSTM networks applied to framewise phoneme classification (Graves and Schmidhuber, 2005) the main difference being that the networks in this work had more than one hidden layer (Abandah, et al., 2015).

The sequence transducer that aligns the input and target sequences consists of two separate RNNs: the input network, which is typically bidirectional, and the prediction network, which must be unidirectional along with a feedforward output network used to combine the outputs of the two RNNs. The two networks are used to determine a separate distribution *Pr(k|t, u)* for every combination of input timestep *t* and output timestep *u*. Each distribution covers the *K* possible labels in the task plus a *b*. Intuitively, the network chooses what to output depending both on where it is in the input sequence and the outputs it has already emitted. For a length *U* target sequence y*, the complete set of TU decisions jointly determines a distribution over all possible alignments between x and y*, from which log Pr(y*|x) can be efficiently determined with a forward-backward algorithm (Graves, 2012).

The one-to-one network was trained with online steepest descent (weight updates after every sequence) using a learning rate of 10^{-3} and a momentum of 0:9 and random initial weights drawn uniformly from [-0.1, 0.1]. The network was stopped at the point of lowest label error rate on the validation set (Abandah, et al., 2015).

3.2 Buckwalter Arabic Morphological Analyzer (BAMA)

Buckwalter analyzer is an Arabic morphological analysis tool provided by the LDC. The analyzer produces all potential morphological analyses, calls them solutions, of a given Arabic word. Each solution is fully diacritized according to the morphological status of that word except for the case ending diacritic (Buckwalter,2004). The BAMA tool consists of the following components:

1. Lexicon: each word in Arabic can be segmented into stem and optional affixes. Prefixes, stem, and suffixes in BAMA have separate dictionaries.

Every word in these dictionaries is available as undiacritized and with diacritized variants, its morphological category, and its English meaning. Examples of prefixes are: b (with), k (as), w (and)..etc. examples of suffixes: y (my/mine), k (your/yours), hm (their/theirs masc. pl.)…etc.

2. Compatibility tables: Three compatibility tables "tableAB","tableAC",and "tableBC" that list compatible Prefix with Stem, Prefix with Suffix, and Stem with Suffix respectively. These tables are implemented using linguistic rules only. Each of the three compatibility tables sets pairs of compatible morphological categories. For examples Prefix category "Al" (the) is listed as being compatible with Stem categories "N" (nouns) i.e; Alkitab (The book).

3. Morphology analysis algorithm: Perl code that makes use of the three lexicon files and three compatibility tables in order to perform morphological analysis and produce the possible solutions for each word.

An example of BAMA analysis is shown in Figure 8. Here, the word يكتبون has three solutions each with different meaning and pronunciation. The word has the present tense character ي as the prefix, the stem كتب , and the sound plural masculine ون as the suffix. The letters and diacritics are written using Buckwalter transliteration, see appendix A. The first solution is يَكْتُبونَ , the second solution is : يُكتَبونَ , and the third is يُكتِبونَ. The last letter usually is not diacritized in BAMA solutions but since the sound plural masculine "جمع المذكر السالم" is in declension of the fatha "مبني على الفتح", BAMA diacritizes the last letter with fatha.

INPUT STRING: يكتبون
LOOK-UP WORD: yktbwn
SOLUTION 1: (yakotubuwna) [katab-u_1]
ya/IV3MP+kotub/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
(GLOSS): they (people) + write + [masc.pl.]
SOLUTION 2: (yukotabuwna) [katab-u_1]
yu/IV3MP+kotab/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
(GLOSS): they (people) + be written/be fated/be destined +
[masc.pl.]
SOLUTION 3: (yukotibuwna) [>akotab_1]
yu/IV3MP+kotib/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
(GLOSS): they (people) + dictate/make write + [masc.pl.]

Figure 8. BAMA analysis for the word يكتبون

BAMA may provide wrong analysis. This could happen due to improper Arabic names for places and companies that are not listed in the lexicons. Also common nouns may be used as people names and you do not need to analyze these words (e.g., Khaled, Arabiyat) among many other possible problems (Maamouri, et al., 2004).

CHAPTER IV

Methodology

4.1 Introduction

The diacritization process of an Arabic text can be divided into two types; morphology dependent and syntax dependent. BAMA morphological analyzer can extract some of the morphological diacritics. The rest of diacritics including the end of the word diacritics are restored using RNN statistical approach.

Our diacritization system undergoes several steps to restore diacritics. The first phase is the training phase which is shown in Figure 9. First we perform data encoding. The data is now prepared in two formats; diacritized target sentences and undiacritized input sentences. Then we used BAMA to extract the morphological analysis of each word of the input sentences. Although we did not use an Arabic text tagger; we can infer partial diacritics from the results list of BAMA output solutions. Also, we use the morphological segmentation to get the word components; prefix, stem, and suffix. A text corrector is used next to fix some issues that come up when using BAMA. After that we use one of the four RNN schemes implemented in this thesis to transcribe the input sentences. Next, the RNN is trained to restore the rest of diacritics. In the second phase, the production phase, we use the testing data and apply all the previous steps except that we use the trained RNN networks to restore diacritics as shown in Figure 10. Post processing corrections are applied to improve the accuracy results. The diacritization accuracy metrics: diacritization error rate (DER) and word error rate (WER) are calculated to be compared to other approaches.

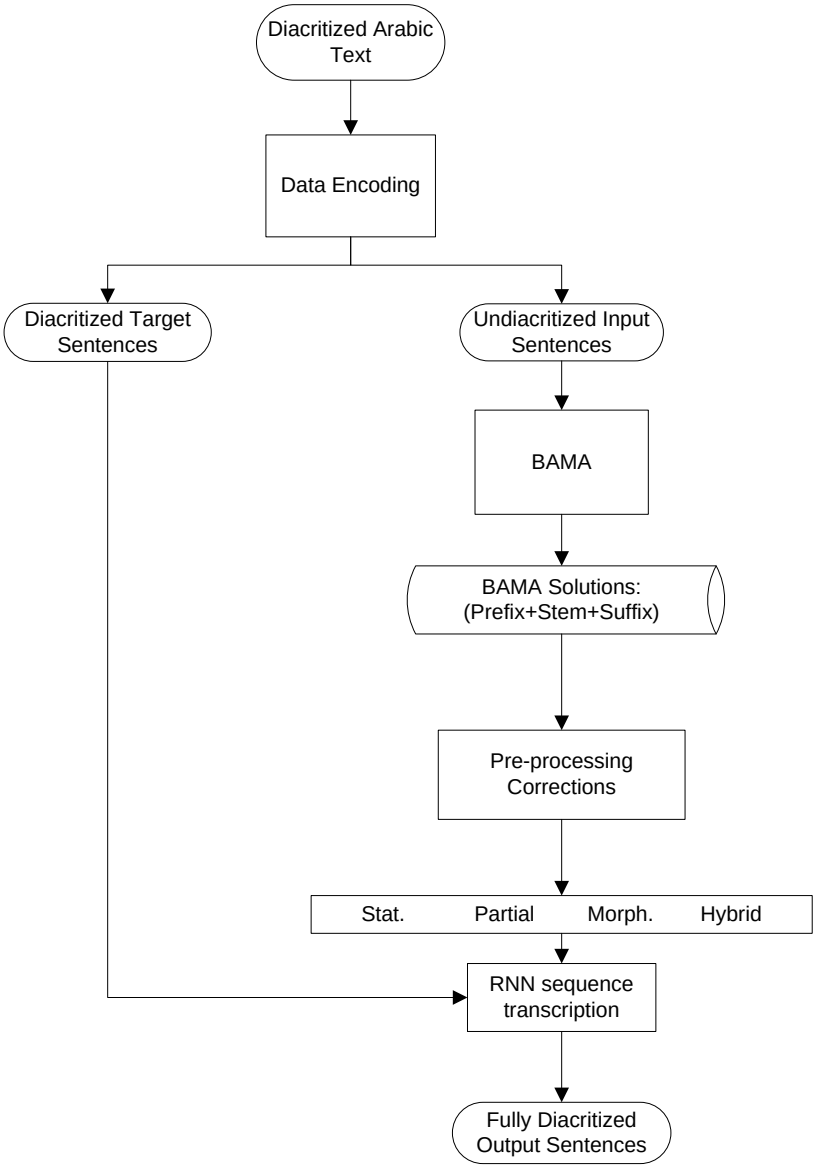

Figure 9. The schematic diagram of the proposed system (Training phase)

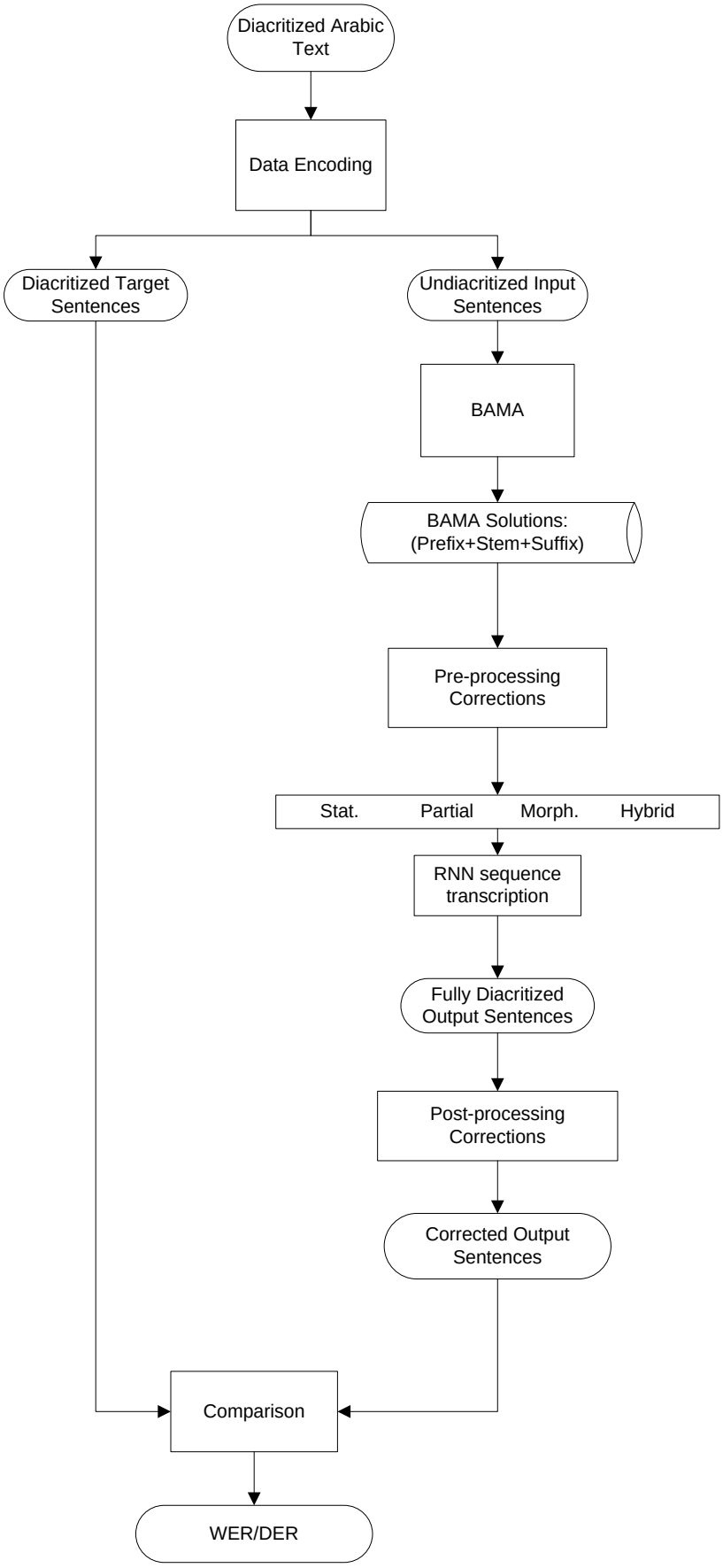

Figure 10. The schematic diagram of the proposed system (Production phase)

4.2 Data

The diacritization schemes we implemented in this thesis are trained and evaluated on the LDC's Arabic Treebank of diacritized news stories – Part 3 v3.2: catalog number LDC2010T08 ((Maamouri, et al., 2004). The LDC ATB3 corpus is a diacritized Arabic text taken from 599 documents from the Lebanese newspaper "AnNahar" of the year 2002 (including case-endings diacritics).

Table 3 below shows some statistics regarding this corpus. Around 40% of the corpus has no diacritics suggesting that the LDC ATB3 corpus is partially diacritized. The percentage of letters with Shaddah and another diacritic is 5.4%. More than eleven words are in a typical sentence. This suggests the possibility of long dependencies between the words. The architecture of deep bidirectional LSTM that is used in this work to transcribe sequences fits these situations.

Table 3. LDC ATB3 statistics

Criterion	**Value**
Size	305 K words
Letters per word	4.64
Words per sentence	11.31
No diacritics	39.8%
One diacritic	54.8%
Two diacritics	5.4%

We choose to use this corpus because the state-of-art approaches in automatic Arabic text diacritization use this Treebank. So, we have established our experiments to allow an equitable comparison of our results directly to theirs.

4.3 Data Pre-processing

A few stages are needed to prepare data for RNN sequence transcription. These stages are illustrated next.

4.3.1 Data Encoding

Each diacritized sentence in LDC ATB3 corpus is placed in a separate line along with the undiacritized version of the same sentence. Comma is used to separate the two versions. This preparation of data is beneficial for supervised learning using RNN.

In Unicode system, diacritics are encoded as separate characters coming after the letter Unicode presentation (see appendix A for Arabic character`s Unicode codes). This encoding technique is called “one-to-many” letter encoding (Abandah, et al., 2015). For example, the word **وَقَّعَ** has the two field record input and target **“وَقَّعَ”,“وقع”** and is encoded as follows:

“ وَقَّعَ”,“وقع”

“0648 0642 0639”,“ 0648 064E 0642 0651 064E 0639 064E”

Dealing with the diacritics as independent characters from letters adds complexity and some difficulties. Abandah et al. (2015) suggested a new method to encode the diacritics with letters using the same character encoding. In this architecture, the input and the target sentences have the same length. This encoding technique is called “one-to-one” letter encoding.

This new encoding is used as follows, Arabic letters have the Unicode codes 0x0621 through 0x063a and 0x0641 through 0x064a (36 letters). We start from the Unicode code for every letter; the most significant 8 bits are removed; the lower 8 bits

are shifted to the left four bits; so we get 12-bit numbers. If the letter is followed by shaddah (0x0651), Bit 3 is set. If the letter is followed by a diacritic other than shadda, then bits 0 through 2 are set using the bit codes shown in Table 4:

Table 4. Binary codes and hexadecimal Unicode's of Arabic diacritics

Diacritic	Unicode	Bit code
No diacritic	-	0000
Fathatan	0x064b	0001
Dammatan	0x064c	0010
Kasratan	0x064d	0011
Fataha	0x064e	0100
Damma	0x064f	0101
Kasra	0x0650	0110
Sukun	0x0652	0111

The following formula is used to calculate a unique code L for the letter with Unicode value l and possible diacritics d_1 and d_2 (bit codes):

$$L = \begin{cases} (l \wedge 0x00ff \ll 4) & no\ diacritics \\ (l \wedge 0x00ff \ll 4) \vee d1 & one\ diacritics \\ (l \wedge 0x00ff \ll 4) \vee d1 \vee d2 & two\ diacritics \end{cases} \quad (15)$$

The previous example is therefore encoded in decimal format as: "0480 0420 0390", "0484 042C 0394"

4.3.2 Using BAMA

BAMA is used by many researchers for Arabic text processing (Vergyri and Kirchhoff, 2004), (Habash and Rambow, 2007), and (Azim et al., 2012). Its capability of generating multiple possible diacritized and morphological analysis solutions for every word makes it quite important in automatic diacritization field.

Let U be the set of undiacritized input words. Then for each word $u \in U$, BAMA gives a list of possible diacritized solutions D_u. For this work, we are interested in the

morphological analysis as a morphological aid scheme. We are interested in the diacritics distributions on the word consonants in the partial diacritization scheme. For the hybrid scheme, we are interested in all information provided by BAMA. If the undicritized word u has n unique solutions, then D_u is the set: $D_u = \{d_{u,1}, d_{u,}2, d_{u,3}, \ldots\ldots, d_{u,n}\}$. This set of solutions is taken after some processing as an input to the RNN diacritizer. Where each *w* word is expressed with triple components so that $w \rightarrow t$ = *(prefix, stem, suffix)*.

The following example illustrates the way we use BAMA to extract partial diacritization and morphological division of words. Take the sentence **الطلاب يكتبون دروسهم لوحدهم** as an example of undicritized input sentence. The BAMA solutions of this sentence are listed in Figure 11.

```
INPUT STRING: الطلاب
LOOK-UP WORD: AlTlAb
  SOLUTION 1: (AlTul~Ab) [Talib_1 ] Al/DET+Tul~Ab/NOUN
    (GLOSS): the + students +
INPUT STRING: يكتبون
LOOK-UP WORD: yktbwn
  SOLUTION 1: (yakotubuwna) [katab-u_1 ]
ya/IV3MP+kotub/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
    (GLOSS): they (people) + write + [masc.pl.]
  SOLUTION 2: (yukotabuwna) [katab-u_1 ]
yu/IV3MP+kotab/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
    (GLOSS): they (people) + be written/be fated/be destined + [masc.pl.]
  SOLUTION 3: (yukotibuwna) [>akotab_1 ]
yu/IV3MP+kotib/VERB_IMPERFECT+uwna/IVSUFF_SUBJ:MP_MOOD:I
    (GLOSS): they (people) + dictate/make write + [masc.pl.]
  INPUT STRING: دروسهم
LOOK-UP WORD: drwshm
  SOLUTION 1: (duruwshm) [daros_1 ] duruws/NOUN+hum/POSS_PRON_3MP
    (GLOSS):  + lessons/classes + their
INPUT STRING: لوحدهم
LOOK-UP WORD: lwHdhm
  SOLUTION 1: (liwaHodhm) [waHod_1 ] li/PREP+waHod/ADV+hum/POSS_PRON_3MP
    (GLOSS): for/to + alone/only + their
  SOLUTION 2: (lawaHodhm) [waHod_1 ] la/EMPHATIC_PARTICLE+waHod/ADV+hum/POSS_PRON_3MP
    (GLOSS): indeed/truly + alone/only + their
  SOLUTION 3: (lawaH~adahum) [waH~ad_1 ]
la/RESULT_CLAUSE_PARTICLE+waH~ada/VERB_PERFECT_SUBJ:3MS+hum/PVSUFF_DO:3MP
    (GLOSS): would have + unite/regularize + he/it <verb> them
```

Figure 11. BAMA output analysis of the sentence **الطلاب يكتبون دروسهم لوحدهم**

For each word, we take the list of solutions and construct an array of prefixes, stems, and suffixes of these solutions. Each one of these entries is an array of characters as well. Figure 12 illustrates how we used the BAMA solutions to build our four schemes. For the Buckwalter transliteration codes of Arabic characters, see appendix A.

INPUT STRING: الطلاب (AlTlAb)			
	Prefix	stem	suffix
SOLUTION 1	Al	Tul~Ab	
INPUT STRING: يكتبون (yktbwn)			
	Prefix	stem	suffix
SOLUTION 1	Ya	kotub	uwna
SOLUTION 2	Yu	kotab	uwna
SOLUTION 3	Yu	kotib	uwna
INPUT STRING: دروسهم (drwshm)			
	Prefix	stem	suffix
SOLUTION 1	-	duruws	hum
INPUT STRING: لوحدهم (lwHdhm)			
	Prefix	stem	suffix
SOLUTION 1	Li	waHod	hum
SOLUTION 2	La	waHod	hum
SOLUTION 3	La	waH~ada	hum

Figure 12. Summary of BAMA solutions for the previous sentence.

To build the first scheme (Statistical), we take the representation of letters with no diacritics. For the previous example the input sequence is:

" AlTlAb yktbwn drwshm lwHdhm "

The second scheme (Partial Diacritization Aid) discards the morphological analysis of the word and uses the persistent diacritics in all solutions of the words to be part of the input sequence. The input sequence in this scheme is:

" AlTul~Ab y kotb uwna duruws hum l waHd hum"

The third scheme (Morphological Aid) benefits from the morphological analysis of the word without the need of diacritization information. The input sequence in this scheme is:

" Al+TlAb y+ ktb+ wn drws+ hm l+ wHd+ hm"

To build the fourth scheme (Hybrid), we take the representation of letters with matched diacritization in all solutions along with the morphological division of each word into its components. If the letter has more than one way of diacritization, we discard the diacritics and use the letter without diacritics. For the previous example the input sequence is:

" Al+Tul~Ab y+ kotb+ uwna duruws+ hum l+ waHd+ hum"

It is worth to mention that usually BAMA doesn't produce the syntactic diacritics. Few exceptions are available for some simple heuristic rules (SHR). Take the word يكتبون for example; the last ending diacritic is Fatha because the sound plural masculine "جمع المذكر السالم" is in declension of the fatha "مبني على الفتح".

4.3.3 Text Correction

BAMA results have some issues that need to be fixed before moving on. We have applied several automatic text correction procedures to overcome these issues.

4.3.3.1 Space and "+" Normalization

There are some extra spaces in BAMA`s results. This causes a serious problem in the use of RNN sequence transcription due to the required one to one

mapping between letters in the input sequence and the target sequence. Therefore, we remove these extra spaces.

For the third and fourth schemes that involve the use of morphological division of words, we used the "+" sign to denote that this is the separation symbol between the word components. We normalize the target sequences to be morphologically divided in a consistent way with the morphologically divided input sequences. This action again restores the one to one mapping between input sequences and target sequences.

4.3.3.2 Extra Alf Removal

We have noticed that some words when morphologically analyzed using BAMA get extra Alf letter. For example, the word "للدول " is analyzed to "ل+الدول ". The same word "لِ+لدُّوَلِ " in the target sentence doesn't have this Alef so we remove this extra letter from the input sequence.

4.3.3.3 Letter Correction

BAMA alters some words letters, increase the number of letters (e.g. كتاب instead of كتب), or decrease the number of letters of some input words (e.g. المر instead of المرتقب). It sometimes also replicates some words in the input sequence many times. We fixe these problems through letter by letter alignment between the input sequence and the target sequence to make sure that they are matched letter-wise.

4.3.3.4 OOV Conversion

Some Arabic proper names or transliterated foreign names cannot be analyzed by BAMA. It returns numbers instead of the names. These numbers indicate the sequence number of the unanalyzed word. This Out Of Vocabulary (OOV) problem is

fixed and the correct names are restored from the target sequences and replaced the printed numbers in input sequences.

4.3.3.5 Target Normalization

After fixing the above problems, we need to morphologically analyze the target sequences in a similar manner to the input sequences for the third and forth schemes. Using the same example illustrated in 4.3.2 the input and target sequences for the third scheme was:

"Al+TlAb y+ktb+wn drws+hm l+wHd+hm", "AlTul~Abu yakotubuwna duruwshmu liwaHodihmu "

And after target normalization will be:

"Al+TlAb y+ktb+wn drws+hm l+wHd+hm", "Al+Tul~Abu ya+kotubu+wna duruws+hmu li+waHodi+hmu "

4.4 Sequence Transcription

The open source software package RNNLIB (Graves, 2008), that is used in this work to transcribe sequences; is built using deep bidirectional LSTM. Many sequence transcription problems have been solved using this architecture and achieved state-of-art results such as handwriting recognition (Abandah, et al., 2014) and speech recognition (Graves, et al., 2013).

We used the “one-to-one” network (Abandah, et al., 2015) to train and test sequences that were prepared using “one-to-one” letter encoding (described in Section 4.3.1), where the input sequences have a one-to-one mapping with the target sequences. The “one-to-many” network (Abandah, et al., 2015) was also tested against the “one-to-

one" and found that "one-to-one" is better in terms of diacritization accuracy. An experimental proof is given in Chapter 5.

4.5 RNN Training Parameters

Zitouni et al. (2006) have suggested a split of the LDC ATB3 corpus that is adopted by many researchers in the automatic diacritization field, (Habash and Rambow, 2007) (Rashwan and Al-Badrashiny, 2011), etc.

This corpus includes 599 documents from the An Nahar News paper. The corpus is split into two sets: training and testing. The training set is the 509 documents of the first ten months of the year 2002. The last 90 documents of this year are the testing set and they are about 15% of the corpus.

This split suggests using the same set as validation and testing rather than using separate validation and test sets as it should. This is not a good practice since it makes the network optimized for the test set.

Although this split is adopted by the follower researcher, we decided to divide the training set into training and validation sets (70% for training and 30% for validation). We kept the last 90 documents for testing only. This split keeps the testing set unseen by the network until the testing step to give a true diacritization accuracy evaluation.

4.6 Data Post-processing

We use the following automatic post-processing techniques to correct some errors in the output sequences of the sequence transcription stage. These corrections have improved the diacritization accuracy.

4.6.1 Letter Correction

The letters in the output sequences after the sequence transcription should be the same in the input and target sequences. Errors could happen to letters in the output sequences. For example, you could have the word "عِلْمٌ" in the output sequence instead of the word "حِلْمٌ" . Although this letter correction step improves the output sequences, it does not really affect the diacritization accuracy calculations because the latter is related to testing the correct diacritics restoration.

4.6.2 Sukun Correction

Some writing styles use the sukun diacritic to indicate that the letter does not have vowel. On the other hand, other styles do not use sukun at all assuming that the letters with no diacritic hold sukun by default. Both styles are correct. So we omit the sukun diacritic from target and output sequences to unify dealing with the sukun. This correction has reduces the diacritic error rate by 6.3% for ATB3.

4.6.3 Fatha Correction

In Arabic orthographic system, the letters Alef, Alef Maksura, and Taa Marbuta are always preceded by a letter that has the Fatha diacritic. If the RNN gives a diacritic other than Fatha, this diacritic is corrected to Fatha using this post-processing step. This correction improves the diacritic accuracy by 1.1% for ATB3.

4.6.4 Dictionary Correction

A dictionary is constructed out of the diacritized words in the training set and is indexed by the undiacritized word. We search for the undiacritized form for an output word. If we find it in the dictionary, we compare this output word with the stored diacritized forms of this word. If we could not find a match, we select the variant that

has the smallest edit distance. If the output is not found in the dictionary, we keep the output diacritics as is. This correction reduces the diacritic error by 1.3% for ATB3 corpus.

The next chapter presents the results of the four schemes that were introduced in Section 4.3.2.

CHAPTER V

Experiments and Results

Before investigating the results of the four suggested schemes to restore diacritics automatically using RNN, we did several preliminary experiments with different training options. These options include the choice of transcription network, the effect of using weight noise distortion, and the size of RNN impact. To the best of our knowledge, this is the first time RNN is used for automatic diacritization. No previous experience on the best network architecture was available (Abandah et al., 2015). So we needed these experiments to tune the RNN to get the best outcome out of it.

5.1 RNN Tuning Experiments

When we did these experiments, the LDC ATB3 corpus was not available for us yet. So we worked with the freely available diacritized books over the internet. We selected parts of Moghny Almohtaj book (794 K words) and "Alahaad Walmathany" book (8 K words) from Tashkila (Zerrouki, 2014) to start experimenting with. The following subsections present the experiments conducted to tune the RNN.

5.1.1 One-to-many versus one-to-one

We have tested the one-to-one network against one-to-many network to decide which network is better for automatic diacritization. In one-to-one network the target sequence has a one-to-one correspondence with the input sequence because the diacritical marks are embedded within the Unicode of the letter. On the other hand, in the one-to-many network, the target sequence is usually longer than the input sequence because the diacritical marks have separate Unicode value from the letter. The internal design of each network on RNNLIB package is illustrated in Abandah et al. (2015).

Figure 13 shows the diacritization error rates (DER) of the two networks using for Moghny Almohtaj book. We have used one hidden layer (250 nodes) on both networks. The one-to-one network achieved lower DER (the proportion of letters with incorrectly restored diacritics). Therefore, we decided to use this transcription method in the following experiments.

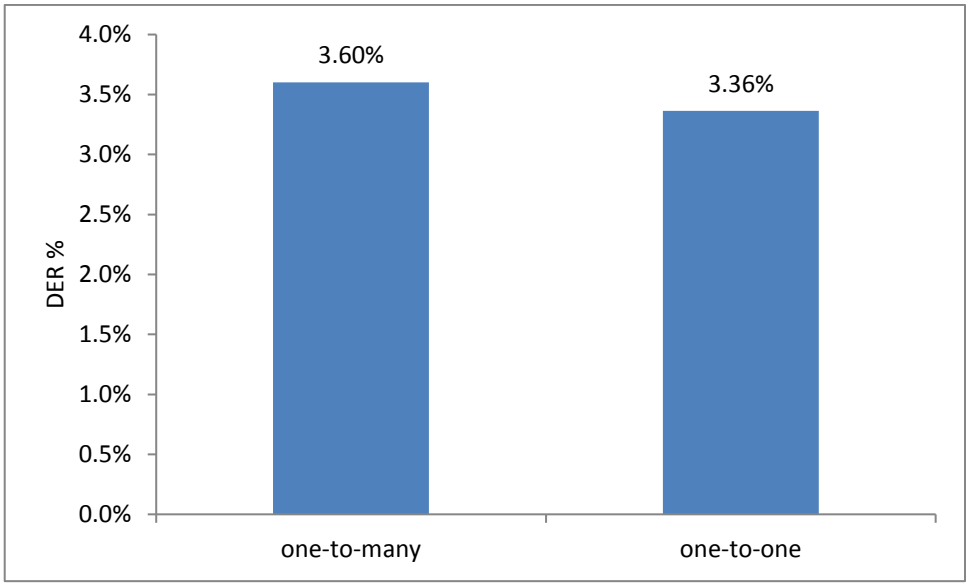

Figure 13. One-to-many vs One-to-one on Moghny Almohtaj book.

5.1.2 Weight noise regularization

Overfitting may occur when training any neural network. This happens when the neural network memorizes the training examples after long training and produces low error rate using the training set. However, when new test data is used, the error rate turnout to be very large. We need to get the neural network to generalize well for new data. Weight noise regularization is one of the methods that is used to solve this problem.

The standard deviation of the noise is selected to be 0.075 as in (Graves, 2011). Figure 14 shows the effect of using weight noise distortion on the classification error rate (rate of number of diacritization errors over total number of symbols) using one-to-

one network of one hidden layer (250 neurons). We used Moghny Almohtaj book for RNN training.

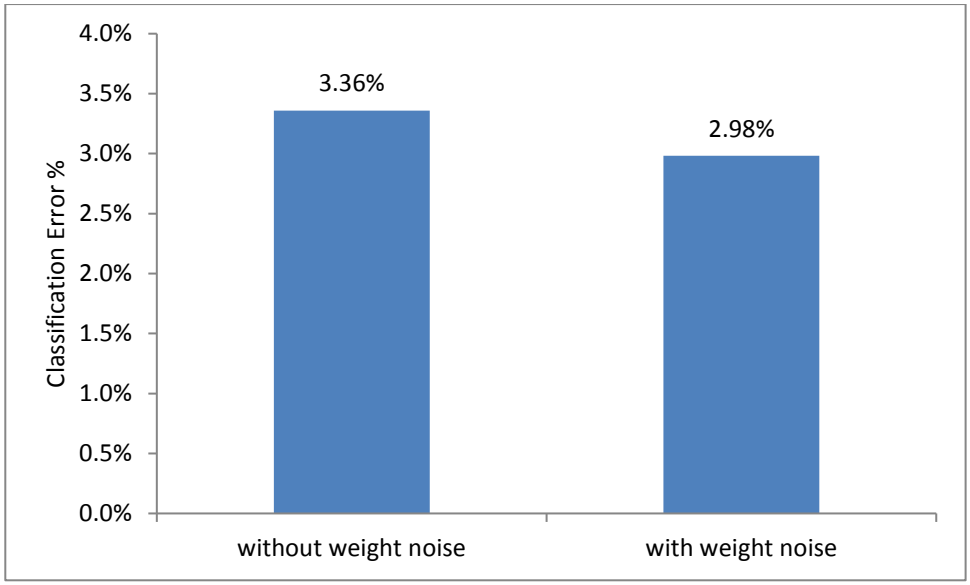

Figure 14. The effect of using weight noise distortion on Moghny Almohtaj book.

We have noticed that using weight noise gives better results. Hence, we use this option in the following experiments.

5.1.3 Network size

The size of the neural network is a function of the number of hidden layers and the number of neurons in each layer. The purpose of this experiment is to determine the size of RNN that gives the optimal accuracy. Figure 15 shows the effect of the number of hidden layers on the classification error rate. In this experiment, each hidden layer has 250 neurons. We found that increasing the number of the layers from one hidden layer to two layers improves the accuracy. In contrast, increasing the number of layers from two to three has a negative impact on the accuracy. Oversized networks could worsen the generalization. So the optimal number of layers for this problem is two hidden layers.

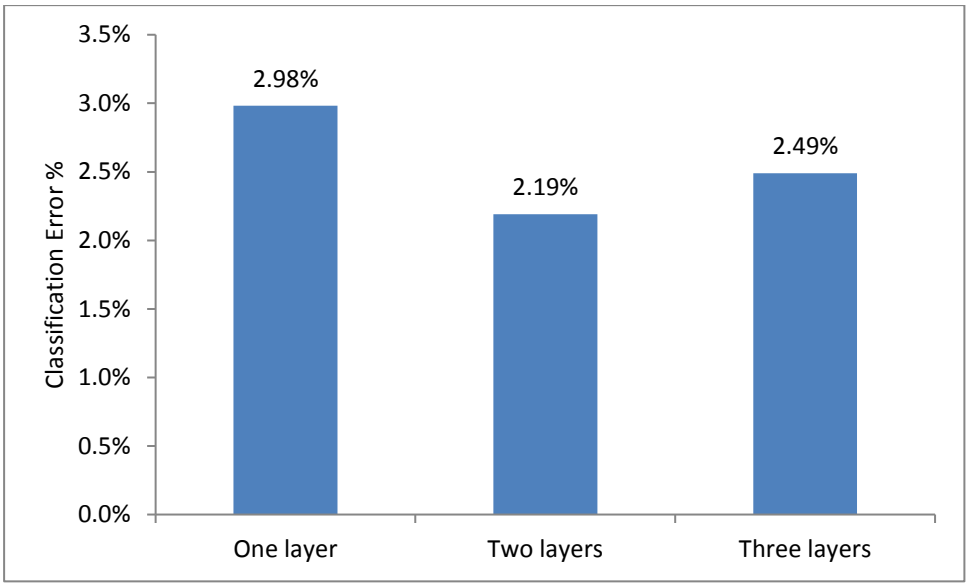

Figure 15. Effect of changing number of hidden layers using Moghny Almohtaj book.

We have also tested the impact of each hidden layer size on the error rate. We found that smaller number of neurons (less than 250) affects the accuracy rate and greater number does not add a significant improvement. Then we tested the effect of reducing one of the hidden layers size on error rate. We found that the error rate increases as the size of the layer decreases no matter if it was the first, middle, or last hidden layer as shown in Figure 16 below.

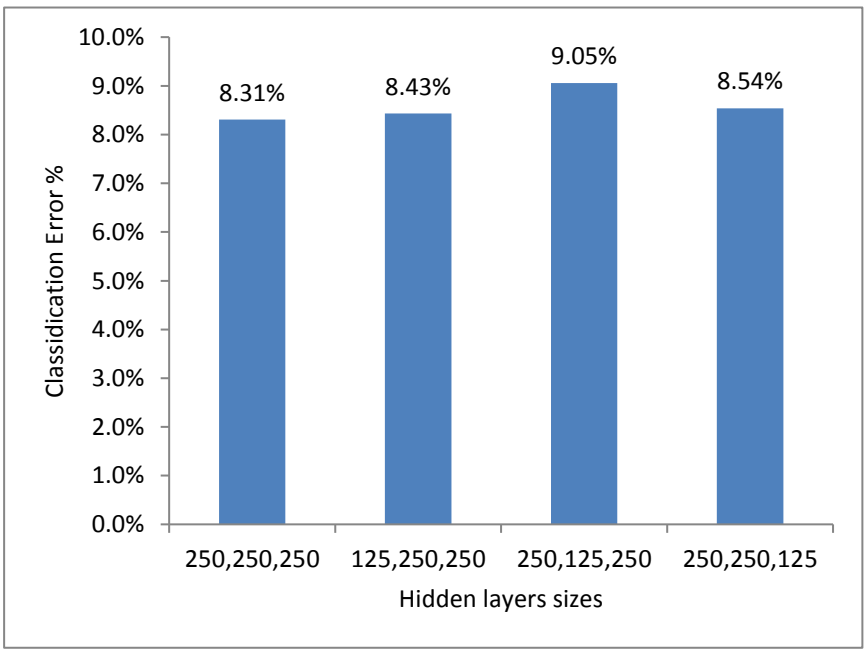

Figure 16. Size effect on each hidden layer using Alahaad Walmathany book.

Therefore, we adopt the one-to-one transcription method with weight noise distortion and two hidden layers each with 250 nodes for all the following experiments.

5.2 Suggested Schemes Results

With reference to the explanation of the four schemes described in section 4.3.2, we present here the results of each of them. Recall that we have implemented four schemes:

1. Statistical Scheme (stat.)
2. Partial Diacritization Aid Scheme (partial.)
3. Morphological Aid Scheme (morph.)
4. Hybrid Scheme (hybrid)

The results presented using two metrics: diacritization error rate (DER) and word error rate (WER). These are the metrics that are used in the literature for performance evaluation where DER represents the proportion of letters with incorrectly restored diacritics, and WER represents the proportion of words that have at least one diacritization error.

We find these accuracy measures for LDC ATB3 under the same conditions mentioned in previous work of (Zitouni et al., 2006), (Habash and Rambow, 2007), (Rashwan and Al-Badrashiny, 2011), and (Said et al., 2013) where:

1. Words, numbers, and punctuators are all used in calculating accuracy.
2. Each letter or digit in a word can hold diacritics.
3. In DER calculation, if the letter holds more than one diacritic (e.g., the letter is hosting Shaddah with some other diacritic) then you need to restore them all or else it will be counted as one error.

4. If the target letter is not diacritized, the same letter in the output sequence is skipped as there is no reference to compare it with.

The results when experimenting on RNN using the four schemes are presented in the Table 5. DER_all and WER_all are the error rates when all diacritization errors are counted. Where DER_ilast and WER_ilast are the error rates when the errors in diacritization the last letter of each word are ignored. The last raw represents the difference between the all-diacritics DER and the ignore-last DER. All schemes are trained using the same division of training and testing set of ATB3 that is described in Section 4.5 and the reported results are the results after automatic post-processing corrections which are described in Section 4.6.

Table 5. Diacritization results of the four schemes using ATB3

Accuracy	stat.	partial.	Morph.	Hybrid
DER_all	3.00	2.74	2.89	2.80
WER_all	10.36	9.66	10.17	9.92
DER_ilast	1.42	1.24	1.36	1.26
WER_ilast	4.52	3.95	4.43	4.07
DER_last	1.58	1.50	1.53	1.54

We have noticed that in all schemes diacritization error rates decrease when we ignore the diacritization error of the last letters of each word. This is expected since restoring the syntactic diacritics on the words ends is much difficult than restoring the morphological diacritics. Recall that the syntactic diacritic is related to the word position in the parsing tree. The WER is also affected by the last letter diacritic since it may not appear on the last letter of the word because it may have a suffix and the syntactic diacritic would appear on the last letter of the stem of that word such as the word "**بكتابِها**". The last raw represents the proportion of last-letter diacritization errors to all letters errors. About 50% of the errors are due to the last letter diacritics.

5.3 Post-processing Contribution

The post-processing procedures improve the diacritization error rates. Table 6 details the contribution of each post-processing correction on DER calculation.

The Sukun correction has the greatest contribution among other techniques. This is predictable since the ATB3 is partially diacritized. Fatha correction contributes with 1.1% in reducing the DER values. It is not a significant value but still is considered a good practice in improving the error rates.

Table 6. Impact of the post-processing techniques on DER reduction.

Technique	Reduction %
Sukun correction	6.3
Fatha correction	1.1
Dictionary correction	1.3
Total	8.7

The dictionary used in ATB3 is created using the diacritized words variants in the training set. The ATB3 corpus is implemented using training set of approximately the first ten months stories of AnNahar newspaper (~258 K words) and a test set using the last two months stories of the same year 2002 (~47 K words). The low contribution of dictionary correction is a logical consequence of having new words and vocabulary in the test set that are not present in the stories of the train set. If we use a bigger dictionary that includes all the words with all of their diacritized variants, we would achieve a better correction contribution out of the dictionary correction.

5.4 Discussion

We found that the second scheme (partial.) has the best results among the three other schemes. It is predictable that it would be better than the statistical approach (the

first scheme) since it adds partial diacritization to the input sequences and this would apparently help the RNN in restoring the rest of the diacritics.

We expected that the fourth scheme (hybrid) would be the best implementation since it benefits from all the linguistic information added to other schemes. Unfortunately, this is not the case. On investigating the problem, we found that the contribution of the morphological segmentation is the reason that worsens the results of the fourth scheme. It is not that the morphological segmentation is a bad practice; it is the way the segmentation implemented that caused the problem. Take the verse " أليس الله بكاف عبده" for instance; if you analyze the word "عبده" using BAMA, you will get the potential solutions shown in Figure 17.

```
INPUT STRING:  عبـد ه
LOOK-UP WORD: Ebdh
        SOLUTION 1: (Eaboduh) [Eaboduh_1]
                    Eaboduh/NOUN_PROP
           (GLOSS):  + Abdo/Abduh +
        SOLUTION 2: (Eabadahu) [Eabad-u_1]
                    Eabad/VERB_PERFECT+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS
           (GLOSS):  + worship + he/it <verb> it/him
        SOLUTION 3: (Eab~adahu) [Eab~ad_1]
                    Eab~ad/VERB_PERFECT+a/PVSUFF_SUBJ:3MS+hu/PVSUFF_DO:3MS
           (GLOSS):  + enslave + he/it <verb> it/him
        SOLUTION 4: (Eabodh) [Eabod_2]
                    Eabod/NOUN+hu/POSS_PRON_3MS
           (GLOSS):  + slave/servant + its/his
```

Figure 17. BAMA analyses of the word عبده

Figure 18 shows the summary of BAMA results regarding the morphological segmentation. In our morphological scheme, we firstly check for the number of segments or morphemes resulted in all solutions. If they were different we return the raw input word without segmentation. So this word is used in input sequence as "Ebdh" instead of being used as "Eabod + hu".

INPUT STRING: عبده (Ebdh)					
	Word Form	Prefix	Stem	Suffix1	Suffix2
SOLUTION 1	Eaboduh (عَبْدُه)	0	Eaboduh	0	0
SOLUTION 2	Eabadahu(عَبَدَهُ)	0	Eabad	a	hu
SOLUTION 3	Eab~adahu(عَبَّدَهُ)	0	Eab~ad	a	hu
SOLUTION 4	Eabodh(عَبْدِه)	0	Eabod	hu	0

Figure 18. Summary of BAMA analysis of the word عبده

We could not do better with the available information. If we used a POS tagger, we would know that it is the fourth solution that should be selected to be the solution of this word. The POS tags are bolded in Figure 17. Incorrect segmentation led to this degradation in the results of schemes three and four.

Nevertheless, it is still better than that the statistical approach since it adds morphological information to the input sequences by segmenting words into morphemes. This practice seems to be beneficial to the RNN in its process of sequence transcription. Figure 19 and Figure 20 demonstrate the error rates DER and WER for the four schemes.

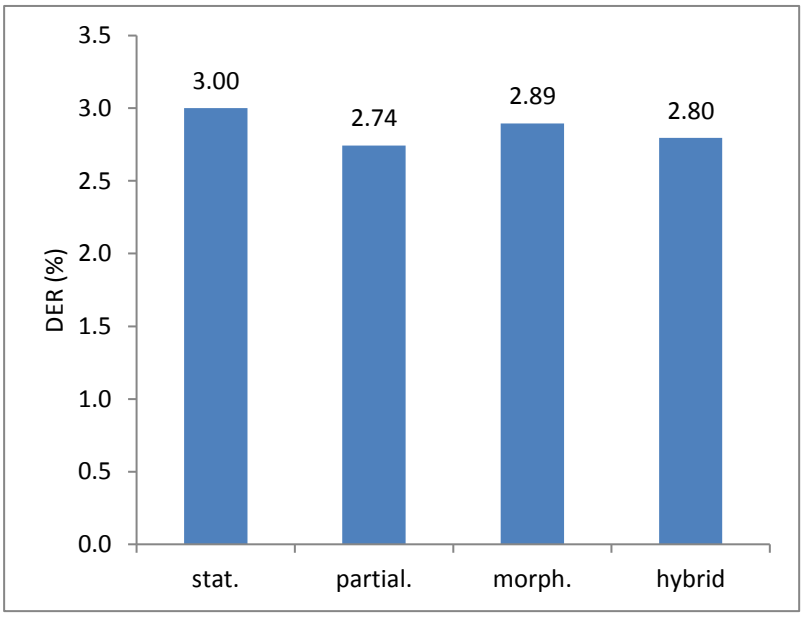

Figure 19. DER results of the four schemes

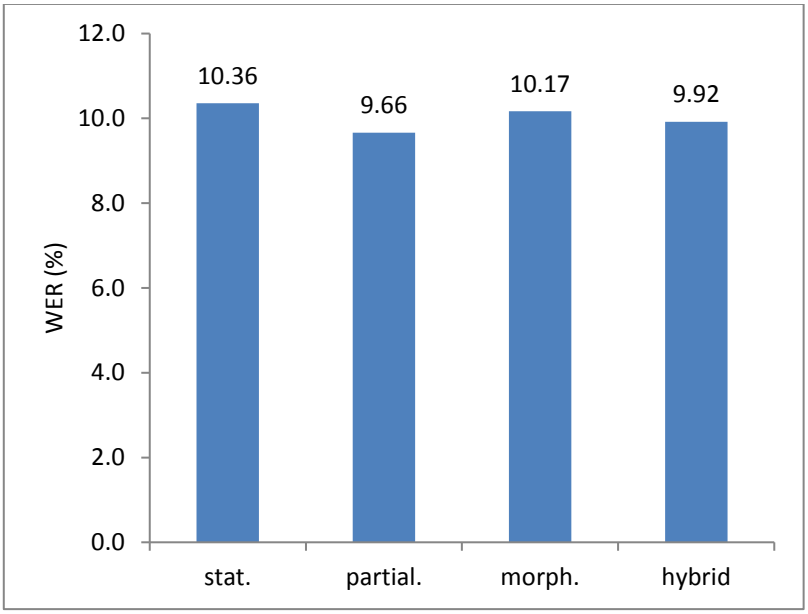

Figure 20. WER results of the four schemes.

However, using the morphological segmentation does improve the classification as Figure 21 shows. This proves that word segmentation into its morphemes got the potential to improve the diacritization accuracy if it were done correctly. We have the intention of using the POS feature in the future to apply correct segmentation of the words. In addition to that, POS is beneficial in selecting partial diacritics of the partial scheme. Refer to the example in Figure 17, if the POS tagging is used and the word عبده was used in a sentence where its role in the parsing tree was verb, then solution 2 and 3 will be picked for comparison (as they are tagged as verbs). The morphemes of these solutions are compared to each other; if they match, then the morpheme is copied as is. If they do not, the mapping stops on the letter where the difference first appears. So the word عبده with verb POS tagging would look like "Eabdahu" in the second scheme and "Eabd+a+hu" in the fourth scheme.

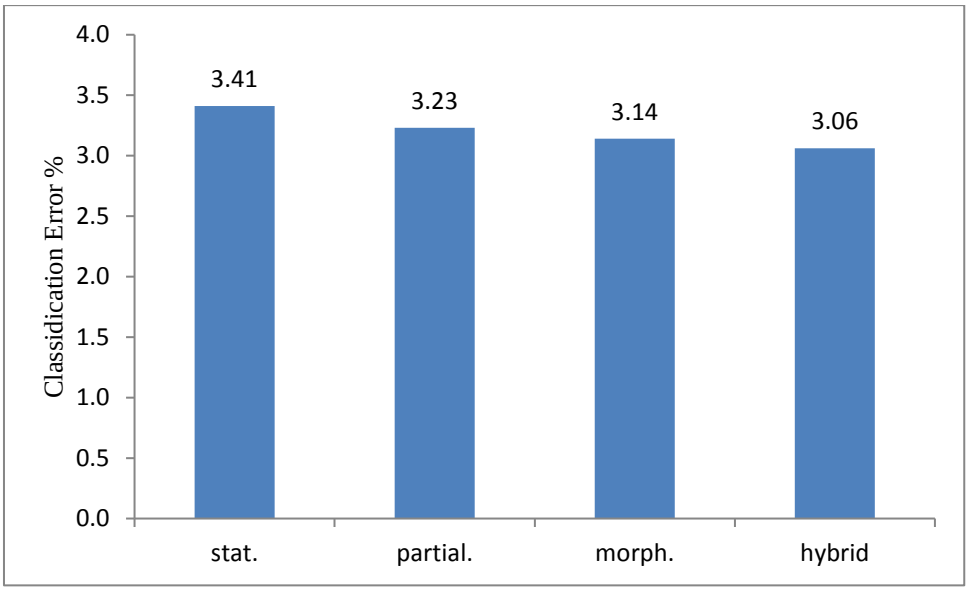

Figure 21. RNN classification error rates for all schemes.

5.4.1 Comparison with state-of-art systems

Table 7 reviews the accuracy results to restore diacritics automatically of the best published researches in the field. Zitouni et al. (2006), Habash and Rambow (2007), Rashwan et al. (2011), Said et al. (2013) are all utilizing hybrid approaches that combine statistical approaches with rule-based approaches. Abandah et al. (2015) use statistical approach based on the deep bidirectional LSTM. All these systems are tested using LDC ATB3 to be comparable with each other.

Table 7. Comparison between our diacritization schemes results with related work.

System	All Diacritics		Ignore Last		DER Last
	DER	WER	DER	WER	
Zitouni et al. (2006)	5.5	18.0	2.5	7.9	3
Habash and Rambow (2007)	4.8	14.9	2.2	5.5	2.6
Rashwan et al. (2011)	3.8	12.5	1.2	3.1	2.6
Said et al. (2013)	3.6	11.4	1.6	4.4	2
Abandah et al. (2015)	2.72	9.07	1.38	4.34	1.34
This work: Statistical scheme	3.00	10.36	1.42	4.52	1.58
Partial scheme	2.74	9.66	1.24	3.95	1.50
Morphological scheme	2.89	10.17	1.36	4.43	1.53
Hybrid scheme	2.80	9.92	1.26	4.07	1.54

As depicted in Table 7, our schemes provide the best results compared to all state-of-art hybrid approaches. The partial scheme achieved the highest scores among the three linguistic-added information schemes and provides 24% DER improvement and 15% WER improvement over the best reported hybrid approach of Said et al. (2013).

Abandah, et al. (2015) achieved 2.72 and 9.07% diacritic and word error rates, respectively. We achieved 3.00 and 10.36% diacritic and word error rates, respectively, using the statistical scheme. Our statistical scheme is exactly the same as Abandah, et al. (2015). The only difference is that Abandah, et al. used the LDC ATB3 corpus split that initiated by Zitouni et al. (2006) and adopted by the follower researchers. This split uses the same set as validation and testing rather than using separate validation and test sets as it should. We kept the last 90 documents of the ATB3 for testing only (previously used as validation and testing sets). This split keeps the testing set unseen by the network until the testing step to give a true diacritization accuracy calculation. This is the interpretation of Abandah, et al. superiority over our system. We are intending to repeat all the experiments again using Zitouni et al. (2006) split of the corpus to provide fair comparisons to other approaches. It is expected that the results of all of our schemes will be enhanced when adopting Zitouni's split.

5.4.2 Error Analysis

In this section, we first inspect the distribution of diacritic errors in words and whether the last letter diacritic is counted or not. Table 8 shows the distribution of errors for all schemes. One% denotes the proportion of the words that have only one error. Two% refers to the proportion of the words that have two diacritization errors. If the word has three or more errors then it is counted under Three+% column.

Table 8. Distribution of word errors for all schemes.

Scheme	Errors per word	One%	Two%	Three+%	Total %
stat.	Last letter correct	27.10	7.80	1.86	36.76
	Error in last letter	52.97	7.47	2.80	63.24
partial.	Last letter correct	26.02	8.32	1.41	35.75
	Error in last letter	55.43	6.32	2.49	64.25
morph.	Last letter correct	26.88	8.50	1.36	36.73
	Error in last letter	53.68	6.76	2.83	63.27
hybrid	last letter correct	26.33	7.56	1.53	35.42
	Error in last letter	55.31	6.51	2.76	64.58

We noticed that the results are close to each other since we are using the same corpus with the same training parameters.

The table shows that in all schemes about 80% of the words ,that contribute in calculating WER, are having one diacritic error, nearly 15% having two errors, and only about 4% are having three or more errors.

Almost more than 63% of the words have error in the last letter. When studying sample of the corpus we found that many words that have error in the last letter (syntactic diacritics) are not really words, in fact they are proper names or foreign transliterated names. So they contribute in increasing the DER and WER percentages.

We also noticed that restoring shaddah is harder than restoring the other diacritics. It is common in some Arabic texts that writers write without diacritics except for shadda. In this case, we need to restore all diacritics but shadda. To find out the effect of having shadda in the input sequences, we ran two experiments using statistical scheme on “Alahaad Walmathany” book. The first has shadda in its input sequences (true shadda), and the second do not and has to predict all diacritics including shadda (predicted shadda). The results are shown in Figure 22. We found that having shadda in input sequences improves the classification error rate by 11%. This is predictable, but

we are sticking with the configuration that treats shadda as one of the eight diacritics that need to be predicted.

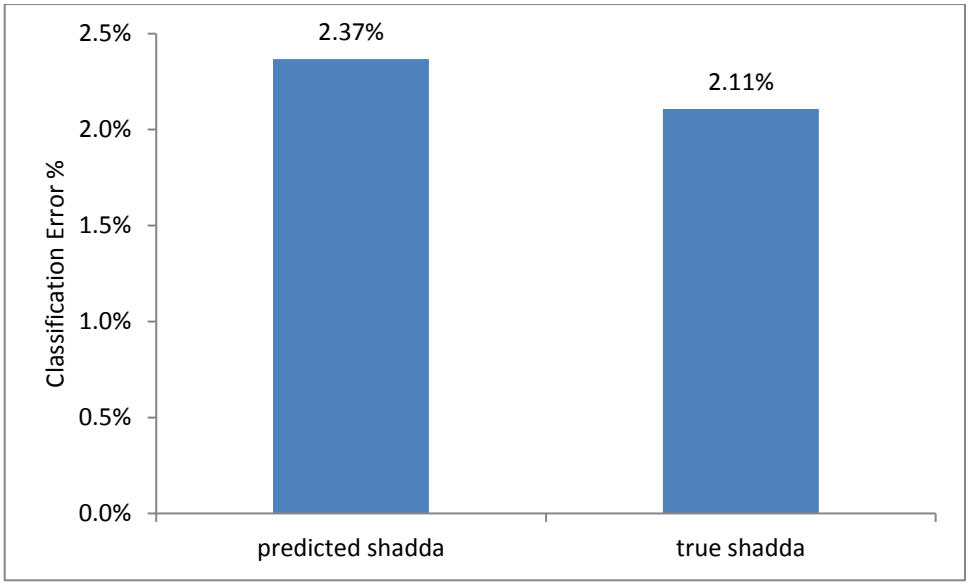

Figure 22. Effect of shadda on error rate.

CHAPTER VI

Conclusions and Future Work

The basic rule in Arabic text diacritization is أشكِل ما أشكَل; i.e., you need to diacritize the words that could cause ambiguity in order to get the correct pronunciation and the intended meaning of the word.

Many researchers in the literature have tried to tackle this problem automatically using rule-based, statistical, and hybrid methods. To the best of our knowledge this is the first time that RNN is utilized to restore diacritics using a sequence transcription network of deep bidirectional LSTM (Abandah, et al., 2015). With the aid of the morphological analysis provided by BAMA, we could add extra linguistic information to the raw input Arabic text. This has proven to be a good practice and it did increase the diacritization accuracy.

As this method has not been used before to restore diacritics, we did several preliminary experiments with different training options to decide the best configuration of RNN network. We found that using one-to-one transcription method with two hidden layers (each 250 neuron) and applying weight noise distortion is the best configuration.

Our diacritization schemes have resulted in a superior outcome on LDC ATB3. About 24% DER improvement and 15% WER improvement is achieved over the best reported hybrid approach of Said et al. (2013).

BAMA solutions are utilized for lexical analyses that provide partial diacritization and morphological segmentation to the input sequences. On the other hand, RNN is used to restore the rest of diacritics especially the last ending diacritic that

depends on the context. The deep bidirectional LSTM memory cells can handle dependences between words in both directions in long sentences.

In this work, we applied some post processing on diacritics such as sukun and fatha corrections and this has improved the results. In future, we could apply other corrections according to some other rules of Arabic text diacritization. For example, a word-initial letter cannot host shaddah or sukun. Tanween diacritics can only be placed on the word-final letter. Also, special diacritization rules can be applied to some letters: ا, آ and ى do not host diacritics; ة, ء and إ do not host shaddah. The letters that precede long vowels ا,و and ي carry a diacritic similar to the vowel, i. e, fatha, damma and kasrah, respectively (Elshafei, et al., 2006). أ , إ , ئ , ؤ and ء represent the glottal stop, but are written in different forms depending on the consonant position in the word. أ could have damma or fatha. إ is always diacritized with kasra. أ, ؤ and ئ are preceded by a diacritic harmonize the consonant of the glottal stop, i. e, fatha, damma and kasrah, respectively.

Our results came consistent with (Azim et al., 2012) and (Rashwan et al., 2011) that segmenting words into morphemes of prefixes, stems, and suffixes are more useful units to restore diacritics than whole words. This segmentation was not utilized in its best capability because we need POS tagger that could assign grammatical part-of-speech tags to words as it is being used in context. Therefore, we will consider using POS tagger in the future so that the selection of a specific analysis out of list of BAMA solution will be more precise and beneficial. Take the following BAMA analysis of the word من in Figure 23. If we are using a POS tagger, we would select the ultimate solution directly. The POS tags are bolded. If the word is tagged as preposition we would select the first solution (مِن) as the appropriate diacritization form. If it is tagged

as pronoun, the solution would be (مَن). The verb form of this word is (مَنَّ) and the noun form is (مَنّ). The POS tagger selects one of these tags according to the context.

```
      INPUT STRING: مـن
      LOOK-UP WORD: mn
           Comment:
             INDEX: P2W48
        SOLUTION 1: (min) [min_1] min/PREP
           (GLOSS): from
        SOLUTION 2: (man) [man_1] man/REL_PRON
           (GLOSS): who/whom
        SOLUTION 3: (man~a) [man~-u_1] man~/PV+a/PVSUFF_SUBJ:3MS
           (GLOSS): bestow/grant + he/it [verb]
        SOLUTION 4: (man~) [man~_1] man~/NOUN
           (GLOSS): grace/favor
```

Figure 23. BAMA analysis of the word من

Many Arabic language computer processing researches including automatic diacritization have been developed into commercial applications. This is for sure is something good but what if these applications were not honest is serving the language? What if they use rules that were morphologically or syntactically incorrect? Even if this was not done on purpose, we should be aware to them. Earlier researches in the literature were done by people who were not even Arabs like Gal in (Gal, 2002) (Vergyri and Kirchhoff, 2004), (Nelken and Shieber, 2005) and others. They tackled the diacritization problem using statistical approaches so they did not need to understand the language first. Although their efforts are highly appreciated, it is our job as Arabic native speakers to develop this research so that we can be sure that it is perfectly handled without any potential manipulation. Arabic is the language of Islam, and serving it is considered as serving Islam. May Allah accepts this work and rewards us on it.

REFERENCES

Abandah, G. Graves, A. Al-Shagoor, B. Arabiyat, A. Jamour, F. Al-Taee, M (2015), Automatic diacritization of Arabic text using recurrent neural networks, **International Journal on Document Analysis and Recognition**, Available on doi:10.1007/s10032-015-0242-2 © Springer-Verlag Berlin Heidelberg

Abu-Hamedeh, Z. (2009), Diacritization marks in the Arabic script and its impact on the level of absorption (experimental study), **Jordan Academy of Arabic No. (57)**, pp. (39) (in Arabic)

Anastasiou, D. (2011). Survey on Speech, Machine Translation and Gestures in Ambient Assisted Living. Proceedings of the **'Métiers et technologies de la traduction: quelles convergences pour l'avenir?'** -Paris.

Azim, A. Wang, X. Sim, K. (2012), A weighted combination of speech with text-based models for Arabic diacritization, Proceedings of the **13th Annual Conference of International Speech Communication Association**, pp. 2334–2337

Azmi, A, Almajed, R (2013), A survey of automatic Arabic diacritization techniques. **Natural Language Engineering**, Available on CJO doi:10.1017/S1351324913000284

Buckwalter, T. (2004), Buckwalter Arabic Morphological Analyzer, v2.0 edn, **Linguistic Data Consortium**, Philadelphia.

Cho, K., van Merrienboer, B., Gulcehre, C., Bougares, F., Schwenk, H., & Bengio, Y. (2014), Learning phrase representations using rnn encoder-decoder for statistical machine translation, **arXiv** preprint arXiv:1406.1078.

Ciresan, D. C., Meier, U., Gambardella, L. M., & Schmidhuber, J. (2010), Deep, big, simple neural nets for handwritten digit recognition, **Neural computation 22(12)**, pp. 3207-3220

El Hihi, S. and Bengio, Y. (1995), Hierarchical Recurrent Neural Networks for Long-Term Dependencies, Proceedings of the **NIPS,** pp. 493-499.

El-Imam, Y. (2004), Phonetization of Arabic: rules and algorithms, Proceedings of the **Computer Speech & Language 18(4)**, pp.339-373.

Elman, J. (1990), Finding structure in time, **Cognitive science**, 14(2), 179-211.

El-Sadany, T. and Hashish, M. (1988), Semi-Automatic Vowelization of Arabic Verbs, Proceedings of the **10th National Computer Conference**, Saudi Arabia, pp. 725-732.

Farghaly, A., & Shaalan, K. (2009), Arabic natural language processing: Challenges and solutions. **ACM Transactions on Asian Language Information Processing (TALIP),** 8(4), 14.

Gal, Y. (2002), An HMM Approach to Vowel Restoration in Arabic and Hebrew, Proceedings of the **Workshop on Computational Approaches to Semitic Languages**. Philadelphia, pp. 27–33.

Gers, F. A., Schraudolph, N. N., & Schmidhuber, J. (2003), Learning precise timing with LSTM recurrent networks, **The Journal of Machine Learning Research**, 3, pp.115-143.

Graves, A. (2012), Sequence transduction with recurrent neural networks, Proceedings of the **ICML Representation Learning Worksop**.

Graves, A. et al. (2013), Speech recognition with deep recurrent neural networks, Proceedings of the **Acoustics, Speech and Signal Processing (ICASSP)**, IEEE International Conference, pp. 6645-6649.

Graves, A. and Schmidhuber, J. (2005), Framewise phoneme classification with bidirectional LSTM and other neural network architectures, **Neural Networks 18(5-6)**, pp. 602-610.

Habash, N. and Rambow, O. (2007), Arabic Diacritization Through Full Morphological Tagging, Proceedings of the **North American Chapter of the Association for Computational Linguistics** (NAACL), pp. 53-56.

Hermans, M. and Schrauwen, B. (2013), Training and analysing deep recurrent neural networks, Proceedings of the **Neural Information Processing Systems,** pp. 190-198.

Hifny, Y. (2012), Smoothing techniques for Arabic diacritics restoration, Proceedings of the **12th Conference on Language Engineering (ESOLEC '12)**, Cairo, Egypt.

Hochreiter, S. and Schmidhuber, J. (1997), Long short-term memory, **Neural computation**, 9(8), pp.1735-1780.

Huang, P. S., He, X., Gao, J., Deng, L., Acero, A., & Heck, L. (2013), Learning deep structured semantic models for web search using clickthrough data, Proceedings of the 22nd **ACM international conference on Conference on information & knowledge management,** pp. 2333-2338.

Kheder, M. (1999), Use of Neural Networks in Arabic Text Transliteration. Proceedings of the 4th **International Conference on Recent Trends in Computer Science Applications & Information Systems,** Philadelphia University Amman, Jordan, pp. 13-14.

Kirchhoff, K., Bilmes, J., Das, S., Duta, N., Egan, M., Ji, G., ... & Vergyri, D. (2002), Novel approaches to Arabic speech recognition: report from the 2002 **Johns-Hopkins summer workshop**, ICASSP'03, Hong Kong, vol.l, pp. 344-347.

Lewis, M. (2009), Ethnologue: Languages of the World, 16th edn. SIL International, Dallas, Tex. Online version: http://www. ethnologue. com.

Maamouri, M., Bies, A., Buckwalter, T., & Mekki, W. (2004), The Penn Arabic treebank: Building a large-scale annotated Arabic corpus, Proceedings of the **NEMLAR conference on Arabic language resources and tools,** Cairo, Egypt, pp. 102-109.

Nelken, R. and Shieber, S. (2005), Arabic Diacritization Using Weighted Finite-State Transducers, Proceedings of the **Workshop on Computational Approaches to Semitic Languages**. University of Michigan, Ann Arbor, pp. 79–86.

Pineda, F. (1987), Generalization of back-propagation to recurrent neural networks. **Physical review letters**, 59(19), 2229.

Rashwan, M. and Al-Badrashiny, M. (2011), A Stochastic Arabic Diacritizer Based on a Hybrid of Factorized and Unfactorized Textual Features, **IEEE Transactions on Audio, Speech, and Language Processing**, vol.19, no.1, pp.166-175.

Robinson, A (1994), An application of recurrent nets to phone probability estimation, **Neural Networks, IEEE Transactions**, 5(2), pp.298-305.

Said, A., El-Sharqwi, M., Chalabi, A., Kamal, E. (2013), A hybrid approach for Arabic diacritization. A Hybrid Approach for Arabic Diacritization. Proceedings of the **Natural Language Processing and Information Systems,** Springer Berlin Heidelberg, pp. 53-64.

Schuster, M. and Paliwal, K. (1997), Bidirectional recurrent neural networks, **Signal Processing, IEEE Transactions** on, 45(11), pp. 2673-2681

Shaalan, K. (2010), Rule-based approach in Arabic natural language processing, **International Journal on Information and Communication Technologies (IJICT)**, Serial Publications 3(3):11–9.

Vergyri, D. and Kirchhoff, K. (2004), Automatic Diacritization of Arabic for Acoustic Modeling in Speech Recognition, Proceedings of the **20th International Conference on Computational Linguistics**. Geneva, pp. 66–73.

Zitouni, I. and Sarikaya, R. (2006), 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), Sydney, Australia, pp.577-584.

Zerrouki, T. (2011), Tashkeela: Arabic vocalized text corpus, Retreived April 5, 2015, from http://aracorpus.e3rab.com/.

Appendix A

Unicode and Buckwalter transliteration of Arabic characters.

Arabic Character	Shape	Unicode	Buckwalter
Arabic Letter HAMZA	ء	U+0621	"
Arabic Letter ALEF with MADDA above	آ	U+0622	|
Arabic Letter ALEF with HAMZA above	أ	U+0623	>
Arabic Letter WAW with HAMZA above	ؤ	U+0624	&
Arabic Letter ALEF with HAMZA BELOW	إ	U+0625	<
Arabic Letter YEH with HAMZA above	ئ	U+0626	}
Arabic Letter ALEF	ا	U+0627	A
Arabic Letter BEH	ب	U+0628	B
Arabic Letter TEH MARBUTA	ة	U+0629	P
Arabic Letter TEH	ت	U+062A	t
Arabic Letter THEH	ث	U+062B	v
Arabic Letter JEEM	ج	U+062C	j
Arabic Letter HAH	ح	U+062D	H
Arabic Letter KHAH	خ	U+062E	x
Arabic Letter DAL	د	U+062F	d
Arabic Letter THAL	ذ	U+0630	*
Arabic Letter REH	ر	U+0631	r
Arabic Letter ZAIN	ز	U+0632	z
Arabic Letter SEEN	س	U+0633	s
Arabic Letter SHEEN	ش	U+0634	$
Arabic Letter SAD	ص	U+0635	S
Arabic Letter DAD	ض	U+0636	D
Arabic Letter TAH	ط	U+0637	T
Arabic Letter ZAH	ظ	U+0638	Z
Arabic Letter AIN	ع	U+0639	E
Arabic Letter GHAIN	غ	U+063A	g
Arabic Letter FEH	ف	U+0641	f
Arabic Letter QAF	ق	U+0642	q
Arabic Letter KAF	ك	U+0643	k
Arabic Letter LAM	ل	U+0644	l
Arabic Letter MEEM	م	U+0645	m
Arabic Letter NOON	ن	U+0646	n
Arabic Letter HEH	ه	U+0647	h
Arabic Letter WAW	و	U+0648	w
Arabic Letter ALEF MAKSURA	ى	U+0649	Y
Arabic Letter YEH	ي	U+064A	y

Arabic Character	Shape	Unicode	Buckwalter
Arabic FATHATAN	ً	U+064B	F
Arabic DAMMATAN	ٌ	U+064C	N
Arabic KASRATAN	ٍ	U+064D	K
Arabic FATHA	َ	U+064E	a
Arabic DAMMA	ُ	U+064F	u
Arabic KASRA	ِ	U+0650	i
Arabic SHADDA	ّ	U+0651	~
Arabic SUKUN	ْ	U+0652	o

التشكيل الآلي للنصوص العربية باستخدام الشبكات العصبونية ذات التغذية الراجعة

إعداد
آلاء خالد رضوان عربيات

المشرف
الدكتور غيث علي عبندة

ملخـــــص

النصوص العربية المستخدمة حاليا في المدارس والجامعات وأماكن العمل وفي الكتب والاعلام غالبا ما تكون غير مشكلة. السبب في هذا هو لتقليل أجور الطباعة. نتيجة لذلك، الكثير من الكلمات تكون مبهمة وذلك لأن نفس الكلمة ممكن أن تقرأ بأكثر من طريقة وممكن أن يكون لها أكثر من معنى. أبناء اللغة العربية يستطيعون استنباط اللفظ الصحيح والمعنى لكلمة معينة من خبرتهم وعلمهم باللغة ومن سياق الجملة لكن الأجانب والأطفال الذين يتعلمون اللغة حديثا لا يستطيعون، بالإضافة لذلك تطبيقات اللغة الحاسوبية مثل تمييز الكلام بشكل آلي والتحويل من نص مكتوب الى كلام منطوق والعديد من التطبيقات الأخرى كلها تحتاج إلى ان يكون النص مشكولا.

عالج الباحثون في هذا المجال مسألة التشكيل بطرق مختلفة. من هذه الطرق تلك التي تعتمد على قواعد النحو والصرف لاستنباط حركات التشكيل المناسبة، وأخرى تعتمد الطرق الاحصائية، وهناك نوع هجين حقق أفضل النتائج لغاية الآن؛ وهو الذي يدمج بين الطرق الاحصائية والمعرفة اللغوية بقواعد النحو والصرف.

تم تقديم مقترح جديد لتشكيل النصوص في هذا البحث بحيث يتم استغلال الشبكات العصبونية ذات التغذية الراجعة في استرجاع حركات التشكيل باستخدام طريقة تحويل المتسلسلات للشبكات العميقة ثنائية الإتجاه ذات وحدات التخزين طويلة الأمد وقصيرة الأمد. أثبتت هذه الطريقة الإحصائية كفاءتها في مجال التشكيل حيث حققت نتائج متفوقة على قريناتها من الطرق في الأبحاث الأخرى. قمنا بعد ذلك ببناء طرق أخرى تعتمد على هذه الطريقة الإحصائية ولكن باضافة معلومات لغوية تساعد الشبكات العصبونية ذات التغذية الراجعة على استرجاع علامات التشكيل بشكل أفضل لنحصل على طريقة هجينة. تم تعزيز نتائج الشبكات العصبونية ببعض الوسائل التصحيحة وهذا ساهم في تحسين دقة التشكيل أكثر فأكثر.

اضافة المعلومات اللغوية ساعد الشبكات العصبونية ذات التغذية الراجعة على تحقيق نتائج هي الأكثر دقة لغاية الآن في مجال أنظمة التشكيل الهجينة. باستخدام كتاب جمعية البيانات اللغوية الجزء الثالث، حقق نظامنا الهجين نسبة خطأ في التشكيل على مستوى الحرف 74.2% ونسبة خطأ على مستوى الكلمة 9.66%. اذا لم نحتسب حركة التشكيل على الحرف الأخير فإن نسبة الخطأ على مستوى الحرف تصبح 1.24% و 3.95% على مستوى الكلمة.