Media Engineering and Technology Faculty
German University in Cairo

Automatic Diacritization for Arabic Language

Bachelor Thesis

Author: Sedki Gabra Sedki Abdelhakim
Supervisor: Dr. Mohamed Elmahdy
Submission Date: 15 May, 2017

Media Engineering and Technology Faculty
German University in Cairo

Automatic Diacritization for Arabic Language

Bachelor Thesis

Author: Sedki Gabra Sedki Abdelhakim
Supervisor: Dr. Mohamed Elmahdy
Submission Date: 15 May, 2017

This is to certify that:

(i) the thesis comprises only my original work toward the Bachelor Degree

(ii) due acknowlegement has been made in the text to all other material used

Sedki Gabra Sedki Abdelhakim
15 May, 2017

Acknowledgments

I would like to express my deepest gratitude and appreciation to all who supported me throughout the course of this thesis. First of all, I would like to sincerely thank my supervisor, Dr. Mohamed El-Mahdy for his continuous guidance, assistance and attention to every small detail. I also thank all my friends and colleagues for the fun and enjoyable time we had. Finally, my deepest thanks and gratitude to my family for their unconditional love, support and encouragement.

Abstract

Most Arabic text nowadays is written without diacritic marks which can be problematic and lead to potential ambiguity. Although this ambiguity can be resolved by native speakers through understanding the whole context, it still can be a source of confusion to non-native speakers and people with some learning disabilities. So this thesis recherches some of the techniques used for automatically restoring those missing diacritics and also proposes a hybrid system to further decrease the error rates of restoring them.

The researched techniques were split into two parts: letter based approaches and word based approaches. Decision Tree, Naïve Bayes, Support Vector Machines and Multi-layer Perceptron were researched as letter based approaches. And in word based approaches, different n-grams models were researched. Those techniques were tested against Taskeela dataset. The results showed that Decsion Tree was the best in letter based techniques and Combined N-grams Models Using Backoff was the best in word based techniques. The proposed hybrid system that combines those two techniques together outperforms both with results in Diacritization Error Rate (DER), Word Error Rate (WER) of 4.35% and 12.6%; and 4.33% and 12.51% when ignoring the last letter.

Contents

Chapter 1

Introduction

1.1 Motivation

Most Arabic text nowadays is written without diacritic marks, this missing diacritics can lead to potential ambiguity. While such ambiguity can be resolved easily from the context by native speakers, it still can be a source of confusion to non-native speakers and people with some learning disabilities. Diacritization is even more problematic for some applications like text-to-speech since the correct pronunciation of Arabic words not only depends on the letters, but also the diacritic marks. All that inspired this thesis to tackle the problem of missing diacritics.

1.2 Main Objective

The main objective of this thesis is to implement and evaluate various models from two categories: letter based approaches and word based approaches to automatically restore diacritics for Arabic text. And also build a hybrid model that combines the best two models from both worlds to obtain the best results of both.

1.3 Structure of the Thesis

Chapter 2 gives an overview of the some related tools, concepts and approaches that are required to fully understand this thesis. After that, the methodology chapter that illustrates the implementation details and flow of execution for the targeted systems and models and also illustrates the hybrid system proposed. Next, the experimental results chapter that mentions the results for each system. Finally, the conclusion and future work chapter that represents a final statement about the results and also suggests some improvements to be done to enhance them.

Chapter 2

Background

2.1 The Arabic Language

The Arabic language is one of the most spoken languages in the world, with over 400 million speakers in the Arab world. It is also one of the six official languages in the United Nations. The Arabic alphabet consists of 28 letters to represent short consonants and long vowels; and some diacritic marks (shortly **diacritics**). Arabic diacritics (**tashkil**) helps in pronunciation of short vowels as well as indicating the length of consonants. The table below 2.1 shows some of the Arabic diacritics and their corresponding Unicode.

Table 2.1: Arabic diacritics

#	Diacritical Mark	Unicode
1	Fatha	U+064E
2	Damma	U+064F
3	Kasra	U+0650
4	Shadda	U+0651
5	Sukun	U+0652
5	Fathatan	U+064B
6	Dammatan	U+064C
7	Kasratan	U+064D

These diacritics act as a phonetical guide which helps in differentiating between words having the same letters but different pronunciation. The table below 2.2 shows examples of Arabic words with different diacritics and their corresponding English meanings.

Table 2.2: The different meanings for words having the same letters but different diacritics

#	Word Without Diacritics	Word with diacritics	The English Meaning
1	العالم	العَالَمْ	The world
2	العالم	العَالِمْ	The scientist
3	كتب	كُتُبْ	Books
4	كتب	كَتَبَ	Wrote
5	كتب	كُتِبَ	To be written
6	ذهب	ذَهَبْ	Gold
7	ذهب	ذَهَبَ	Went

They also differentiate the role of the target word in its sentence, e.g.,

- In English: The student goes to his school
- In Arabic: يذهب الطالبُ الي مدرسته

In the English sentence, the subject (**the student**) is determined by its position in the sentence. But, in Arabic, the role of words are determined by the diaciritcs of last letters. Here, the "Damma" on الطالبُ indicates that it is the subject. Although differentiating the role of any word can be done by understanding its context without the need of diaciritcs, using diacritics will make it a trivial process especially for non-native speakers of Arabic.

Despite the importance of these diacritics, they are often optional and the majority of the Arabic text nowadays is written without them leaving the job of differentiating between words to the experience of the readers themselves. To solve this problem, several Machine Learning (ML) and Natural Language Processing (NLP) techniques have been developed to restore those diacritics automatically which in turn can help in many applications. An overview of some of those applications is given below.

Text-to-speech

A Text To Speech (TTS) synthesizer is a computer-based system that should be able to read any text aloud [7]. With TTS the computer can convert the text into spoken voice in a way that is nearly identical to the human voice. This process can help visually impaired people read digital text very easily. However, a good TTS algorithm can be very hard to develop especially with languages like Arabic that rely on diacritics. So a good algorithm must be implemented to restore those diacritics before attempting to TTS an Arabic text.

Part-of-speech Tagging

Part Of Speech Tagging (POST) is the process of giving a word in a text a corresponding tag depending on its position and role in the text and its neighboring words, e.g., verb, noun, adjective, etc.

Word Sense Disambiguation

Word Sense Disambiguation (WSD) is the process of finding an appropriate meaning for an ambiguous word in a given context [3], e.g., the word مال in مال غصن الشجره means the verb "lean". But, in this sentence: معي مال , it means "money". This process could benefit greatly from removing the extra ambiguity from words that share the same letters but not the same pronunciation (diacritics).

Machine Translation

Machine Translation (MT) is the process of translating a text from one language to another by using a software. And when translating from or to Arabic, the accuracy of the process can be improved through the diacritization of the text before attempting to MT it.

2.2 Machine Learning

ML is a subfield in computer science that is concerned with developing algorithms that enable computers to learn without any explicit or forced programming from an external programmer. These algorithms adapt and evolve with each new input; allowing to solve a wide range of problems where finding a strict algorithm can be considered impossible. E.g., spell checking, face recognition [4] or playing games like Go [17]. ML algorithms can be classified into three categories: supervised learning, unsupervised learning and reinforcement learning.

2.2.1 Supervised Learning

In supervised learning, the desired output for each given input already exists in a set which can be referred to the **training set**. Then the main target is to develop a model that can extract all necessarily features from the input and map them to the output so as to be benefited later to predict the most probable output to a given input outside the this training set [1]. And depending on the type of the output, supervised learning can be classified into two subcategories:

- **Classification**: The output consists of finite number of discrete categorizes or labels. An example of a classification problem would be predicting the gender of people as a function of some properties of their names.
- **Regression**: The output consists of one or more continuous variables. An example of a regression problem would be the prediction of the temperature as a function of the location.

2.2.2 Unsupervised Learning

In unsupervised learning, there is no training set, so the goal is to find the relations that map the inputs to the outputs. This type of learning can be used as an intermediate step for extracting the features in some datasets before using a supervised learning algorithm [1].

2.2.3 Reinforcement Learning

In reinforcement learning, the computer interacts with the environment and takes actions; and depending on the taken actions a reward or punishment can then be given. This type of feedback helps the computer to evolve in order to take better actions in the future that minimize the punishment and maximize the reward [1].

2.3 Natural Language Processing

NLP is an area of research that explores the interaction between computers and human languages (know as **nature languages**). This type of interaction can then be used in such a way to understand and analyze the natural language text or speech in order to solve problems related to it [6]. NLP can be used in many applications in different fields of study. An overview of these applications is given below.

2.3.1 Text Summarization

Text summarization is the process of reducing a full text document and create a concise summary that keeps only important information in that document. This process can help researchers go through hundreds of documents to decide which ones are relevant to their fields of study and worth reading. It also helps authors make a concise abstract of their work that is faithful to the original material provided [12].

2.3.2 Speech Recognition

Speech recognition is the field of computational linguistics that develops models and techniques which capable of converting the human speech into text [13].

2.3.3 Artificial Intelligence

Artificial Intelligence (AI) is a field of study that tries to build intelligent entities as well as understand them, especially intelligent computer programs. These intelligent entities are the ones that try to mimic the human brain in dealing with the environment to take actions in such a way that increases their chances of success with each repetition. [15].

2.4 Related Tools and Classifiers

This section is an overview of the the ML and NLP tools and Classifiers used throughout the thesis. It also discusses the toolkits that provide them.

2.4.1 Toolkits

Throughout the thesis, two toolkits are used: Scikit-learn and Natural Language Toolkit (NLTK).

- **Scikit-learn**: A free and open source project that provides simple and efficient tools for processing large corpora and provides a summary of useful information (**data mining**).
- **NLTK**: Another free, open source and community driven project that provides easy-to-use interfaces to over 50 corpora. It also provides tools for text processing, such as tokenization, stemming, tagging and parsing.

The next sections discuss some of those provided tools and techniques required by this thesis.

2.4.2 Tools

Tokenization

Tokenization is the process of dividing a string into substrings depending on a given criteria, e.g., splitting text on whitespace and punctuations, thus providing a way to differentiate between words and punctuations.

Regular Expression

Regular Expression (RE) is one of the most used tools in computer science in general and NLP field in particular. It is heavily used in NLP algorithms for detecting patterns in strings (e.g, detecting prefixes and suffixes). These patterns match some strings and do not match others which helps in operations like filtering, searching or replacing parts of strings with new ones.

Text Normalization

Text normalization is the process of finding a canonical form for the input text to simplify the processing of the text later on since the input is guaranteed to be solid and consistent even if it comes from different sources.

Stemming

Stemming is the process of returning a given word to its root by removing the excessive letters from it. It is worth mentioning that stemming does not guarantee the correct root for every given word, instead it guarantees that related words will have the same root. E.g, Arabic words like الطالب , الطالبه or الطلاب will have the same root طلب .

2.4.3 Classifiers

A classifier is not more than a simple function that maps the input feature vectors $x \in X$ (extracted from the **training set**) to the output labels $y \in 1, ..., C$, where X is the set of all features (**features space**) [11]. Below is an overview of some of those classifiers.

Naïve Bayes Classifier

Naïve Bayes Classifier (NBC) is one of the probabilistic classifiers that tries to find the most likely label for a given input by estimating the likelihood for each label, and then choosing the label with the greatest likelihood value. The likelihood of a label is calculated by estimating the prior probability of that label in the dataset, which in turn can be estimated by the frequency of that label in the dataset.

$$\text{Prior Probability of x} = \frac{\text{Occurrences of x}}{\text{Size of the Dataset}}$$

Then each feature will vote against this prior probability to get the likelihood, i.e., reducing the prior probability of a label by multiplying it with the probability that an input value with this label will have that feature [5].

For example, given a dataset of words, in which there are 50% of letters having 'FATHA'. So the prior probability of 'FATHA' is 50%, however if only 10% of the letters having 'FATHA' are ending words then the likelihood of a letter ending a word to have 'FATHA' is reduced from 50% to just 5%.

The mathematics behind NBC is as follows: given a set of features $X = x_1...x_n$ and set of labels Y, then the probability of a label $y \in Y$ given the probability of X is $P(y|x_1...x_n)$.

From Bayes' theorem [16]:

$$P(A|B) = \frac{P(A,B)}{P(B)}$$

So

$$P(y|x_1...x_n) = \frac{P(x_1...x_n, y)}{P(x_1...x_n)} \tag{2.1}$$

And by using the chain rule:

$$\begin{aligned} P(x_1, x_2, x_3...x_n, y) &= P(x_1|x_2, x_3...x_n, y) * P(x_2, x_3...x_n, y) \\ &= P(x_1|x_2, x_3...x_n, y) * P(x_2|x_3...x_n, y) * P(x_3...x_n, y) \\ &= P(x_1|x_2, x_3...x_n, y) * P(x_2|x_3...x_n, y) * ... * P(x_n|y) * P(y) \end{aligned} \tag{2.2}$$

The Naïve Bayes assumption states that features are independent from each other, so equation 2.2 can be rewritten as:

$$\begin{aligned} P(x_1, x_2, x_3...x_n, y) &= P(x_1|y) * P(x_2|y) * ... * P(x_n|y) * P(y) \\ &= P(y) * \prod_{i=1}^{n} P(x_i|y) \end{aligned} \tag{2.3}$$

Finally, by substituting in equation 2.1,

$$P(y|x_1...x_n) = \frac{P(y) * \prod_{i=1}^{n} P(x_i|y)}{P(x_1...x_n)} \tag{2.4}$$

NBC uses equation 2.4 to calculate the probability for each label. Then it chooses the label with the highest probability to indicate the most likely label for a specific set of features.

$$\textbf{Predicted label} = argmax \frac{P(y) * \prod_{i=1}^{n} P(x_i|y)}{P(x_1...x_n)} \tag{2.5}$$

It worth mentioning that $P(x_1...x_n)$ is constant in the calculations, so it can be removed without affecting the predicted label.

Decision Tree

A decision tree is a flowchart where each node is a **decision node**, i.e., a node that represents a test on a feature; and each bottom node, know as **leaf node**, represents the most likely label for the corresponding test value [5]. The process of choosing a label for a given features set is as follows:

One of the features is tested at the beginning of the decision tree, known as **the root**. And depending on the value of the test, one of the branches of the root is chosen. Then, at the arrival of the next node, the same process is repeated until a leaf node is reached. And that leaf node represents the most likely label for the given input.

Construction of Decision Tree

The first step for constructing a decision tree is to build **decision stump** for each feature in the training set. A decision stump is a decision tree with a single node that tests a single feature and classifies the training set based on that test. Then, the best decision stump is chosen. There are several ways to select the best decision stump. The simplest way is to choose the one with the highest accuracy. The accuracy of the decision stump is the ratio between the number of the correct guessed labels to the total number of guessed labels.

$$Accuracy = \frac{\text{Correct labels}}{\text{Total number of labels}}$$

Once the best decision stump is chosen, the accuracy of each leaf node is tested, the one that does not achieve a sufficient accuracy is replaced by another decision stump trained only on the subset of the training set that is selected by the path from the root to the that leaf. This process is repeated until all the leaf nodes have sufficient accuracy or a threshold on the size of the subset of the training set is reached to avoid **overfiting** it, i.e., constructing a decision stump on this subset will result in noise rather than an underlying relationship that can be estimated by its accuracy. A simple representation of the decision tree is shown in figure 2.1 below.

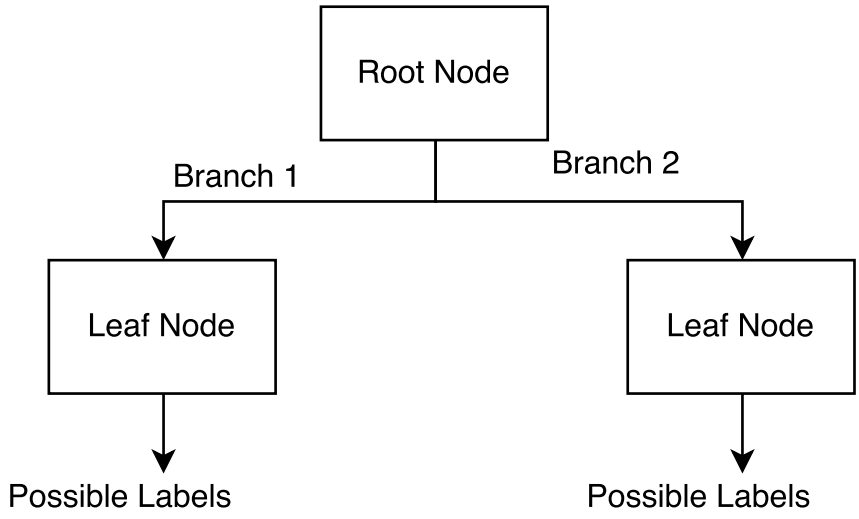

Figure 2.1: Decision tree representation

Support Vector Machines

Support Vector Machines (SVM) is one of the supervised learning classifiers which can be used for solving both classification and regression problems [8].

Given a training set of examples that belongs to one of two classes, SVM will represent those examples as points in n-dimensional space, where n is the number of features in

each example. Then, these points will be divided by a proper hyperplane. In case of having multiple hyperplanes that separate the points, the one that represents the largest separation or margin between the two classes is chosen. So the hyperplane is chosen so that the distance from that plane to the nearest point is maximized. After that, any new example is mapped into the same n-dimensional space and its category is decided according to which group of points it lies on (which side of the hyperplane it lies on).

Figure 2.2 below shows a representation of SVM used to classify a set of examples each containing 2 features (X and Y) into 2 categories (A and B). Two hyperplanes ($h1$ and $h2$) can classify the points, but with different margins. So $h1$ is chosen since it has the largest margin between the two categories.

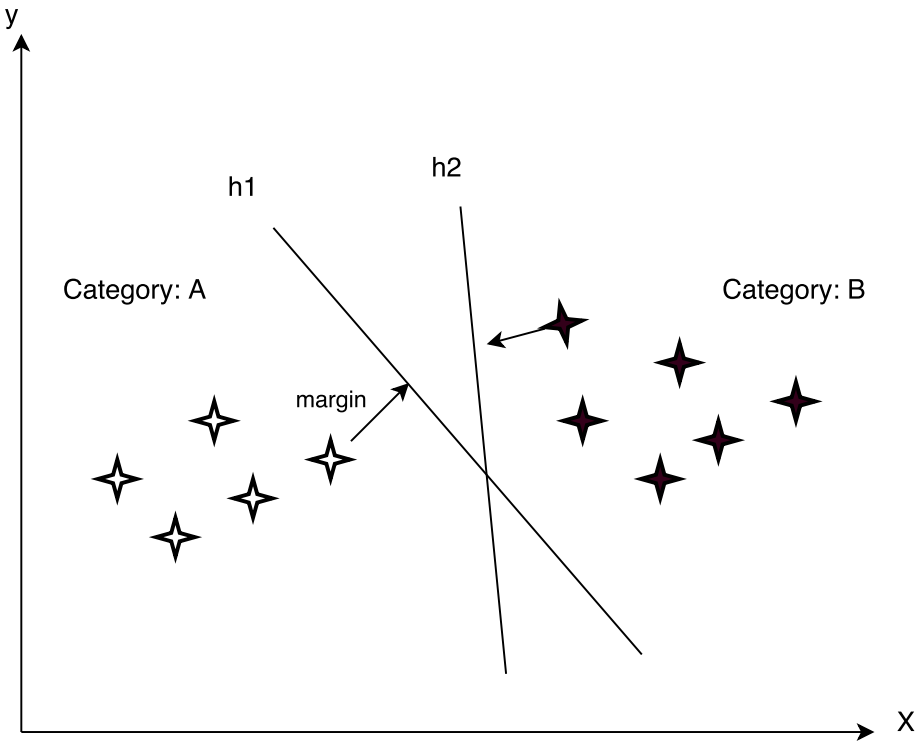

Figure 2.2: SVM representation for classifying a set of examples into 2 categories

Multi-layer Perceptron

Multi-layer Perceptron (MLP) is one of the supervised learning algorithms that tries to map the input to the output by a learning a function $f(\cdot) : R^m \rightarrow R^n$, where m is the number of dimensions for the input and n is the number of dimensions for the output. And this function is obtained by training on the given dataset [9].

Given a set of features $X = x_1, x_2, ..., x_n$ and a target y, MLP can learn a non-linear function that maps this set of features to the target by constructing a directed graph of nodes or **neurons** with multiple layers, with each layer fully connected to the next one,

except for the input nodes. The first layer is the input layer, consisting of a set of neurons representing the input features. Then, one or more hidden layers that transform the values from the previous layer with a weighted linear summation $w_1x_1 + w_2x_2 + ... + w_mx_m$ and a non-linear activation function like the hyperbolic tan function. The last layer is the output layer that transforms the values from the previous layers into output values. Figure 2.3 shows a representation of MLP using one hidden layer.

Learning in MLP

Learning is done by adjusting the connection weights after each new input by using backpropagation. Backpropagation has two phases: propagation and weight update. In the first phase, each new input is propagated forward through the layers, until it reaches the output layer. Then, the output is compared with the expected output and an error value is obtained. After that, The error value are propagated backwards (henceforth the name backpropagation) to calculate a local error value for each neuron depending on its contribution to the output. In the second phase, those error values are used to update the connection weights accordingly.

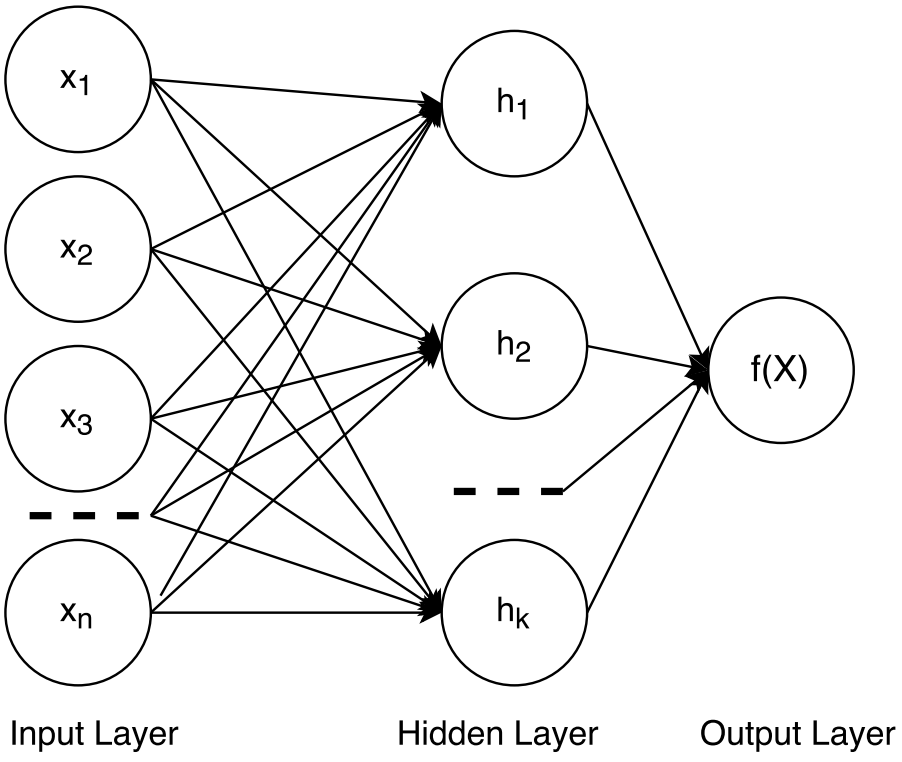

Figure 2.3: MLP with one hidden layer

N-grams

N-gram is a contiguous sequence of n-items extracted from a text. These n-items differ according to the application. E.g., some applications require these n-items to be n-words,

other applications require n-letters and so on [10].

N-gram with $n = 1$ is referred to as **unigram**, with $n = 2$ as **bigram** and with $n = 3$ as **trigram**. When n is greater than 3, n-gram is referred to by the value of n directly, e.g., when $n = 4$ it is called **four-gram**. It is worth mentioning that the total number of possible n-grams for a specific input is always $(s - n - 1)$ where s is the number of units in the input (depending on the type of items each n-gram contains). Table 2.3 below represents the possible unigrams, bigrams and trigrams for the Arabic sentence يذاكر الطالب دروسه

N-grams are used for estimating probabilities of next items in a given sequence. So for a sequence of words $w_1, w_2, ...w_{n-1}$, the probability of the next word w_n is $P(w_n|w_1, w_2, ...w_{n-1})$. This conditional probability can be computed directly by equation 2.6 below.

$$P(w_n|w_1, w_2, ...w_{n-1}) = \frac{P(w_1, w_2, ...w_{n-1}, w_n)}{P(w_1, w_2, ...w_{n-1})} \tag{2.6}$$

However, counting the occurrences of a sequence of words in a given corpora to compute its probability is not feasible in some cases. So the chain rule of probability is used to compute the probability of a sequence of words by chain of conditional probabilities:

$$\begin{aligned} P(w_1, w_2, ...w_{n-1}, w_n) &= P(w_1)P(w_2|w_1)....P(w_n|w_1, w_2, ...w_{n-1}) \\ &= \prod_{i=1}^{n} P(x_i|w_1^i) \end{aligned} \tag{2.7}$$

Where w_1^i is the sequence $w_1, w_2....w_i$

Substitute back in equation 2.6 to get:

$$P(w_n|w_1, w_2, ...w_{n-1}) = \frac{\prod_{i=1}^{n} P(x_i|w_1^i)}{\prod_{i=1}^{n-1} P(x_i|w_1^i)} \tag{2.8}$$

It worth mentioning that the considered history of words depends on the size of the n-gram. E.g., The bigram model approximates the probability of a word given all the previous words $P(w_n|w_1^{n-1})$ by using only the conditional probability of the preceding word $P(w_n|w_{n-1})$

Table 2.3: N-grams example

n	Description	#1	#2	#3
1	Unigrams	يذاكر	الطالب	دروسه
2	Bigrams	يذاكر الطالب	الطالب دروسه	
3	Trigrams	يذاكر الطالب دروسه		

2.5 Prior Work

This section is an overview of the some of the prior work that investigated the problem of automatic Arabic diacritization.

Abandah et al. (2015), [2] used bidirectional Long Short Term Memory (LTSM) network in order to exploit long range context in both directions. LTSM is a special kind of Recurrent Neural Networks (RNN) that is capable of learning the long term dependences which traditional RNN can not. They used samples from ten books of the Tashkila collection and the Quran to test this approach. It worth mentioning that this approach is a pure machine learning, so it did not use any morphological or syntactical analysis. Their approach according to their paper reduced DER by 25%, WER rate by 20%, and the last letter diacritization error rate by 33% over the best published results.

Table 2.4 below shows the reported error rates when all diacritization errors are counted and when the diacritization errors of the last word letters are ignored; and the diacritization error rate of the last letter alone.

Table 2.4: Diacritization results of the LTSM system.

Corpus	All Diacritics		Ignore Last		
	DER	WER	DER	WER	DER-Last
Tashkila	2.09%	5.82%	1.28%	3.54%	0.81%
Quran	3.04%	8.70%	1.98%	5.82%	1.06%
All	4.71%	15.29%	3.07%	10.23%	1.64%

Rashwanand et al. (2009) [14] used a hybrid system of two layers to automatically diacritize raw Arabic text. The first layer tries to decide the most likely diacritics for a given word by choosing from a full-word diacritizations the one with the highest probability using lattice search and n-grams probability estimation. And if the word is out of vocabulary then it goes to the second layer.

The second layer factorizes the word into 4 morphological parts: prefix, root, pattern and suffix where the root is the base from which the word is derived, prefix is the extra letters added to the beginning of the root, suffix is the extra letters added to the end of the root and pattern is the weight of the word against the verb فعل meaning ”do”. Then it uses n-grams and A* lattice search to pick the most likely diacritizations sequence for the given word.

They trained and tested their system by using a dataset of 750K words collected from numerous domains. Table 2.5 below shows the morphological and syntactic diacritization accuracy of the hybrid system when trained on the training-set of different sizes.

Table 2.5: Morphological and syntactic diacritization accuracy of the hybrid diacritizer

Training corpus size	Morphological error	Syntactical errors
128k	9.2%	21%
256k	7.9%	18.7%
512k	6.5%	16.8%
750k	7.0%	16.0%

Zitouni et al. (2006) [18] used a maximum entropy system that combines a wide array of lexical, segment-based and POST features. Lexical features include the neighboring letters to the current one and its neighboring words as well. Segment-based features are the features that split a given Arabic word into three segments or parts: prefix, stem and suffix (refer to section 2.4.2 about stemming). POST features contains the tags of each part of the word.

They used the Arabic Treebank database (LDC's Arabic Treebank Part 3) to test and train their system And reported a DER of 5.1%, Segment Error Rate (SER) of 8.5%, and WER of 17.3%. Where DER is the error rate for each diacritic independently from other diacritics, WER is the error rate for the restored diacritics of each word and SER is the error rate for the restored diacritics of each segment.

Chapter 3

Methodology

This chapter is a discussion about the datasets used in this thesis; their usage, size and and partitioning. It also discuses the different approaches used to solve the problem of automatic diacritization for Arabic text classified under three sections: letter based approaches, word based approaches and hybrid approach.

3.1 Datasets

The datasets are the bread and butter of any classifier because they are used for the purposes of training and testing. To train and test the various systems in this thesis, an accurate dataset was required, i.e, a dataset that contains fully diacritized Arabic text with sufficient number of words. So we settled on Tashkeela; a dataset that contains 75 million of fully diacritized words mainly from 97 books that diverse from classical to modern Arabic language. Some of the 97 books (mostly Islamic classical books) were chosen to represent the dataset of each tested classifier. Table 3.1 below shows a sample of the books in Tashkeela collection and the number of words and letters they contain.

Table 3.1: A sample of books from Tashkeela collection

#	Book Name	Number of letters	Number of Words
1	نخبه الفكر	9,188	2,114
2	متن سلم الوصول الي علم الاصول	12,001	2,807
3	نهايه الرتبه الظريفه في طلب الحسبه الشريفه	73,406	16,828
4	معين الحكام فيما يتردد بين الخصمين من الاحكام	443,084	107,933
5	مؤطأ مالك	554,706	147,572

3.2 Letter Based Approaches

This section discusses the approaches that label only one letter at a time with the most probable diacritic mark depending on its features. A workflow of those approaches is depicted in figure 3.1 below and a further explanation is given in the next sections.

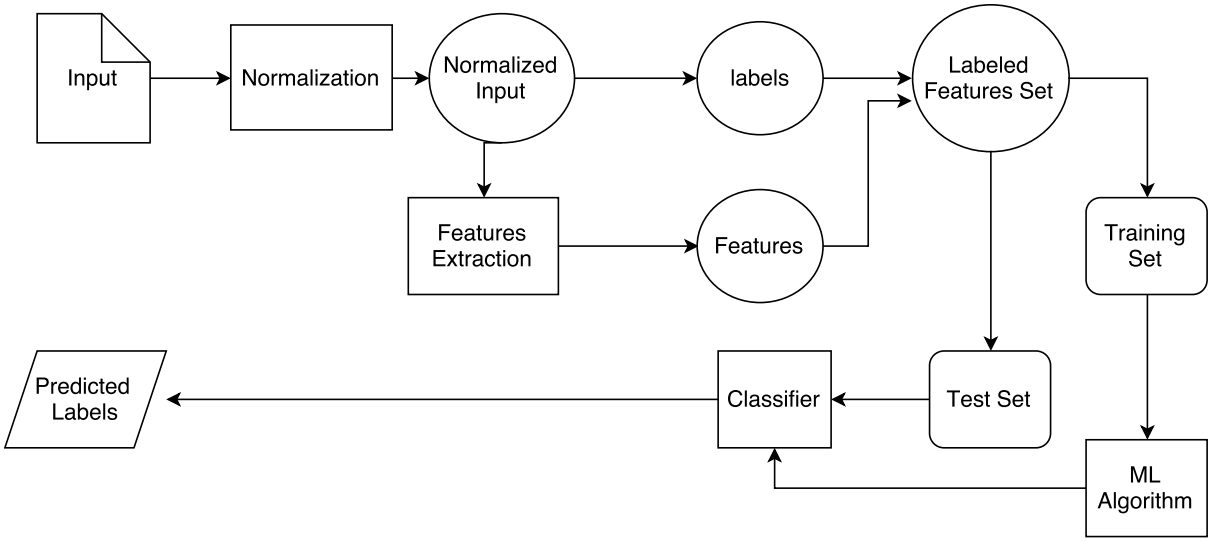

Figure 3.1: System workflow for letter based approaches

3.2.1 Input

The input in letter based approaches is 10 books from Tashkeela collection (see section 3.1) encoded using 8 Bits Unicode Transformation Format (UTF-8). UTF-8 is used to encode letters and 8 means it uses 8-bit blocks to represent a letter. The total number of blocks needed to represent a letter varies from 1 to 4. Any Unicode letter can be encoded using UTF-8, so it is ideal for encoding Arabic input.

3.2.2 Normalization

After reading the input text, normalization (see section 2.4.2) is done on the raw text to solve some problems and assure consistency.

Normalization is done on three steps:

- Filtering out words that contain any English letters or numbers using regular expression (see section 2.4.2).

- Removing dangling diacritics, i.e., diacritics that do not have a preceding Arabic letter.
- Considering only the valid combination of diacritics, i.e., a combination of diacritics that consists of two parts: 'SHADDA' as the first part and any diacritic except 'SHADDA' as the second part, e.g, 'SHADDA' + 'FATHA'.

It is worth mentioning that the Arabic diacritics considered by this thesis are show in table 2.1.

3.2.3 Features Extraction

The features extraction and labeling is done on different steps as shown on figure 3.2 below.

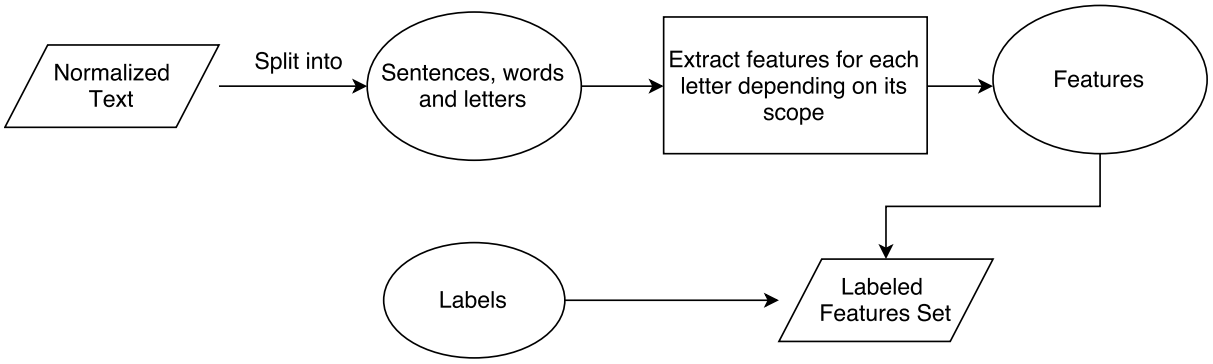

Figure 3.2: The steps of features extraction and labeling

- Splitting the normalized input text into sentences using **sentence tokenizer**. This toknizer is a simple model that returns a list of sentences from a text by splitting on some punctuation marks (full stop, question mark, exclamation mark and colon.
- And for each sentence, the features are extracted for each letter depending on its scope, i.e., the sentence and word it occurred in.
- The last step is labeling the extracted features. This step is done by pairing the extracted features with the diacritic mark of the letter from which they are extracted to form a pair. Then, adding that pair in a set called **labeled features set**.

A further explanation about the nature of those features is given in the next section.

3.2.4 Features

Features are a simplified representation of the input data that try to catch all the relevant characteristics that distinguish one input from another. They can differ from one classification problem to another even for the same input data. And for the considered letter based approaches, the features are selected for one letter in a given context (its word and sentence) according to the following table 3.2:

Table 3.2: The features used in the letter based approaches

#	Feature Name	Description
1	c	Current letter
2	n1	Next letter in sentence, ' ' if not found
3	b1	Previous letter in sentence, ' ' if not found
4	n2	2nd next letter in sentence, ' ' if not found
5	b2	2nd previous letter in sentence, ' ' if not found
6	n3	3rd next letter in sentence, ' ' if not found
7	b3	3rd previous letter in sentence, ' ' if not found
8	n4	4th next letter in sentence, ' ' if not found
9	b4	4th previous letter in sentence, ' ' if not found
10	n5	5th next letter in sentence, ' ' if not found
11	b5	5th previous letter in sentence, ' ' if not found
12	l	Length of the word
13	s	First letter in the word
14	e	Last letter in the word

These features contribute to the final accuracy of the classifier but not equally. Some of them are irrelevant and do not make any contribution to the accuracy or affect the accuracy in a negative way, so they were cut of from the features set. Those features are: length of the word (l), first letter of the word (s) and last letter of the word (e).

3.2.5 Training and Testing the Classifiers

Before training and testing the classifiers, the labeled features set is split into two parts:

- 90% of the labeled features set is used as a **training-set** for training different ML models.

- The remaining 10% is used as **test-set** to test the and evaluate the classifiers.

Then a machine learning model is chosen to be trained by the training-set to obtain a classifier. The choice of those models depends on the nature of the problem itself. And for the problem discussed in this thesis, four models from two toolkits were trained:

- NBC model from NLTK toolkit.
- SVM model from scikit-learn toolkit.
- MLP model from Scikit-learn toolkit.
- Decision Tree model from Scikit-learn toolkit.

Refer to section 2.4.3 for more information about those classifiers. After training each model, it is tested against the test-set. For each letter in the test-set, the predicted label (diacritic) is compared with actual label of that letter.

3.3 Word Based Approaches

Word based approaches are the ones that diacritize the whole word at once based on its context of words. Those approaches use n-grams models to diacritize words (refer to section 2.4.3 about n-grams). Figure 3.3 below shows a system workflow for word based approaches and a further explanation is given in the next sections.

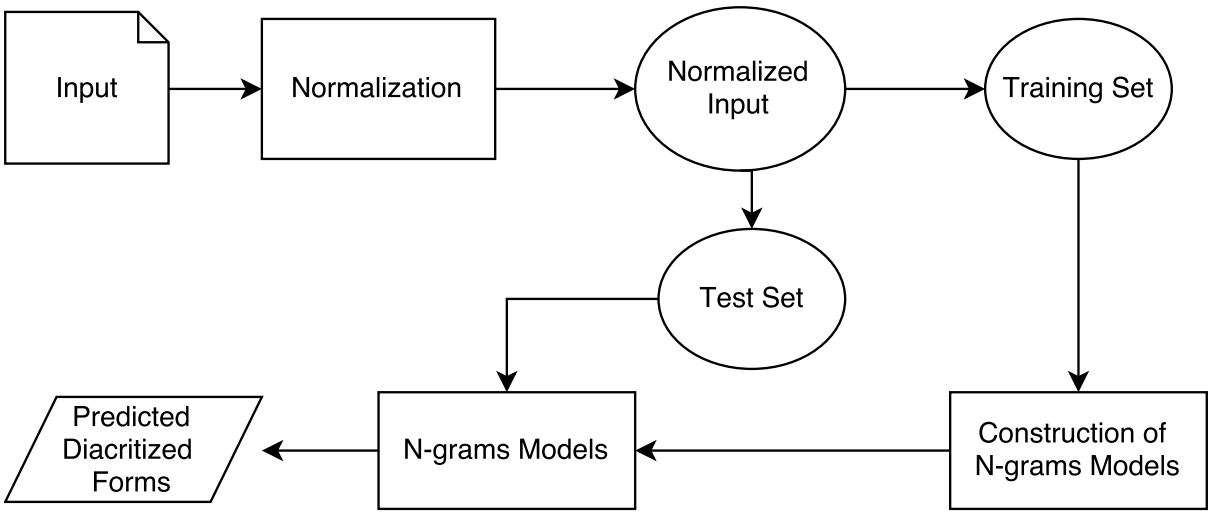

Figure 3.3: System workflow for word based approaches

3.3.1 Input

The input for word based approaches is 16 books from Tashkeela collection encoded using UTF-8 (see section 3.2.1 for more information about UTF-8).

3.3.2 Normalization and Dataset Splitting

Normalization is done similar to letter based approaches (refer to section 3.2.2 about normalization) and the dataset is split into two parts: training set (90%) and test set (10%).

3.3.3 Construction of N-grams Models

Having the training set in hand, several n-grams models were constructed: unigrams, bigrams and trigrams. The n-grams were constructed as list of pairs. Each pair consists of two parts:

- First part is a history of each word in the training set. The length of the history depends on the type of the n-grams being constructed.
- Second part is a list of the possible diacritized forms for the target word accompanied by the frequency of each diacritized form.

3.3.4 Testing N-grams Models

The n-grams models were tested individually to estimate the diacritics for a given word depending only on its history of previous words. E.g., given the word الطالب in the sentence يذاكر الطالب , it will be tested in any of those models as follows:

- The history of the word will be searched for in the target model. If the history is found, then the diacritized form with the highest frequency is chosen. Otherwise the word is reported as Out Of Vocabulary (OOV).
- The frequency of the chosen diacritized form is tested against a threshold value. If it has a value that is at least the threshold value, then it is reported as the most probable diacritized form. Otherwise the word is reported again as OOV.
- Compare the reported diacritized form (if any) with the actual diacritized form from the test set.

The n-grams models also were tested together using **backoff** to estimate the diacritics for a given word by searching through the n-grams models with smaller n under the condition that the word is OOV in the n-grams with bigger n. A word is considered OOV with respect to an n-grams model if its history is not contained in that n-grams model or contained with a frequency strictly less than **threshold** value.

OOV words arise from the fact that, all Arabic words cannot be contained in a single dictionary since Arabic is a morphologically rich language, where new words can be easily created be combining other words. Using backoff with n-grams models ensures that the n-grams model with most reliable information has the advantage to diacritize words, and thus helps to decrease the overall error. Figure 3.4 shows the backoff process from the n-grams with higher n to the ones with less n.

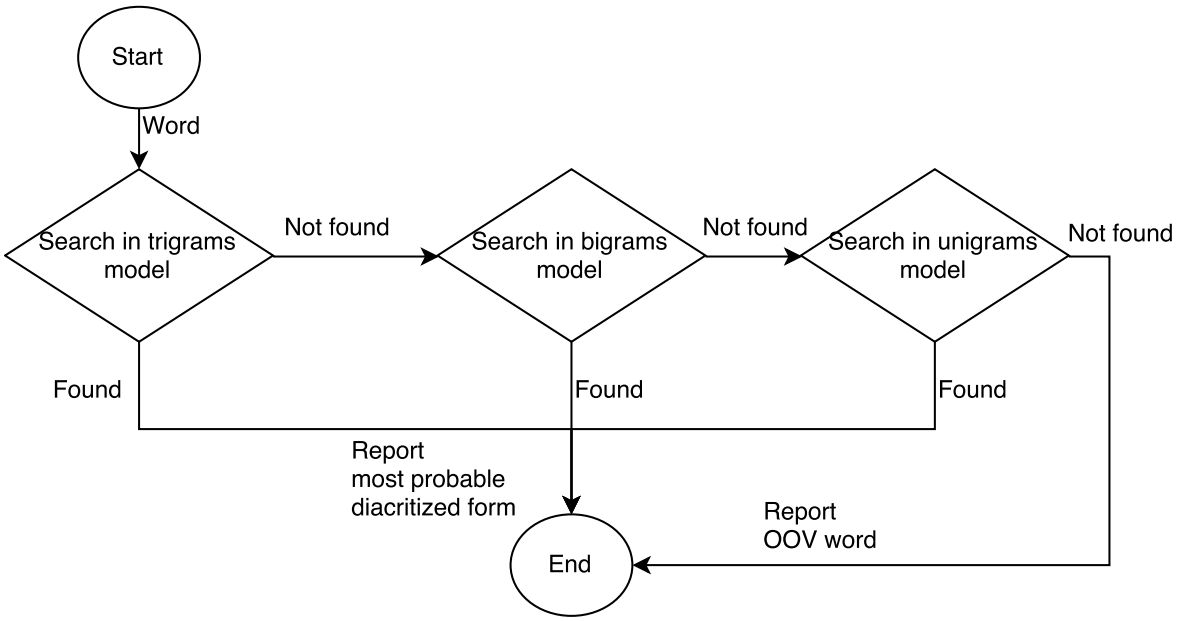

Figure 3.4: The steps of searching the n-grams models using backoff

3.4 Hybrid Approach

The proposed approach in this thesis is the hybrid system. This hybrid system try to combine the best of the two worlds: letter based approaches and word based approaches to be benefited from both worlds. The proposed system consists of two layers.

The first one, is a word based model that uses n-grams models with backoff to try to diacritize a given word by using its history of previous words, if it is not an OOV word then n-grams models will output the most probable diacritized form for the whole word at once. Otherwise, it goes to the second layer.

The second layer, is a letter based model that uses a decision tree to try to diacritize each letter in the word with the most probable diacritic. Then combines these letters and diacritics to from a full diacritized form for the word which has the most probable diacritics.

The diacritized from whether it comes from the first layer or the second is tested against the actual diacritized form obtained from the test set.

Chapter 4

Experimental Results

This chapter discusses the results of running various approaches to automatically diacritize Arabic words. Those approaches are categorized in 3 sections: letter based approaches, word based approaches and hybrid approach. It also discusses the various evaluation metrics and limitations encountered while implementing and evaluating those approaches.

4.1 Evaluation Metrics

A very important part of any system is how it is evaluated and compared to other systems. This section discusses the various evaluation metrics used in this thesis.

- WER: The error rate when comparing the predicted diacritized form for each word as a whole with the actual diacritized form. The predicted diacritized form is considered incorrect if it has at least one incorrect diacritic mark.
- DER: The error rate when comparing the predicted diacritic mark for each letter independently with the actual diacritic mark.
- In Vocab WER: Similar to WER, but it is calculated after excluding the OOV words from consideration.
- In Vocab DER: Similar to DER, but it is calculated after excluding the OOV words from consideration.

4.2 Letter based Approaches

This section mentions the results of running various letter based classifiers against the test-set. Table 4.1 below shows the DER of each classifier with last word letters and

without last word letters (with ignoring last letters completely from each word in the testing set).

Table 4.1: Results of letter based classifiers

#	Classifier Name	DER with Last Letter	DER Without Last Letter
1	NBC	37.16%	32.24%
2	Decision Tree	8.85%	7.01%
3	SVM	29.16%	24.99%
4	MLP	12.08%	8.87%

- From the above table, it is clear that decision tree gives the best results, so it is used as the second layer in the hybrid system.
- The DER without last letter did not improve much over DER with last letter in decision tree (about 1.86%), while the other classifiers showed an improvement of about 3% to 5%. This small improvement is due to the fact that decision tree managed to predict some diacritics for last word letters while the other classifiers failed, so the gap between the two error rates became smaller in decision tree.

4.3 Word based Approaches

This section mentions the results of running various n-grams models as word based classifiers against the test-set. The 3 tables below represent the results for each individual n-grams model and for the 3 models using backoff at 3 different *threshold* values.

Table 4.2: Results of n-grams models when $threshold = 0$

Metric	Unigrams	Bigrams	Trigrams	Combined Models Using Backoff
WER	22.88%	31.31%	62.45%	13.07%
In Vocab WER	22.14%	8.56%	6.42%	12.24%
DER	8.8%	31.14%	63.24%	5.7%
In Vocab DER	7.42%	3.17%	2.68%	4.27%
Words Misses	0.94%	25.01%	59.99%	0.94%

Table 4.3: Results of n-grams models when $threshold = 50$

Metric	Unigrams	Bigrams	Trigrams	Combined Models Using Backoff
WER	28.06%	37.95%	64.35%	16.88%
In Vocab WER	20.08%	5.78%	4.99%	11.29%
DER	19.52%	39.6%	65.51%	12.35%
In Vocab DER	6.9%	2.28%	2.25%	4.02%
Words Misses	10.04%	34.31%	62.57%	6.35%

Table 4.4: Results of n-grams models when $threshold = 100$

Metric	Unigrams	Bigrams	Trigrams	Combined Models Using Backoff
WER	30.8%	40.24%	64.83%	18.44%
In Vocab WER	19.19%	5.4%	4.86%	10.87%
DER	24.52%	41.96%	65.97%	14.86%
In Vocab DER	6.68%	2.16%	2.21%	3.9%
Words Misses	14.47%	37%	63.11%	8.58%

- It is clear from the above tables that increasing the threshold value decreases In Vocab WER and In Vocab DER since it filters out less frequent diacritized forms. Thus, increasing the probability of returning a correct diacritized form for each inquired word. However, it increases WER and DER since the total number of those returned forms is decreased because more and more words will be considered OOV because they do not meet the threshold value.
- Although the combined n-grams models using backoff shows a slight lag behind trigrams and bigrams models in both In vocab WER and In Vocab DER, it has far better results in both WER and DER since it has fewer Words Misses compared to the other models. Words Misses is the ratio between the number of OOV words and the total number of words in the test set.

4.4 Hybrid Approach

The best classifier from letter based approaches is combined with the best model from word based approaches to build a hybrid system. This hybrid system is built to increase

the overall accuracy and reduced the error rate of both words and diacritics in comparison of any single model tested by this thesis. The best classifier in letter based approaches is decision tree, but the best model in word based approaches cannot be determined by simply comparing the results since there is not any n-grams model that has all the best results in all metrics. So several hybrid systems were implemented and the best combination was selected to represent the proposed hybrid system. Table 4.5 below highlights the results of those hybrid systems.

Table 4.5: Results of the hybrid system with different n-grams models at different threshold values

N-grams Model	Threshold:0		Threshold:50		Threshold:100	
	WER	DER	WER	DER	WER	DER
Unigrams	22.43%	7.48%	24.2%	7.99	24.94%	8.17%
Bigrams	17.05%	5.7%	18.82%	6.27%	19.34%	6.47%
Trigrams	23.75%	8.12%	24.01%	8.22%	24.08%	8.25%
Combined Models Using Backoff	**12.6%**	**4.35%**	14.57%	4.93%	15.18%	5.08%

It is now highlighted that the best hybrid system is the one that combines the decision tree and the combined n-grams model using backoff at $threshold = 0$. Table 4.6 below shows the results of running this hybrid system against the test-set in two cases: with last letter and without last letter.

Table 4.6: Results of the hybrid system

Last Letter	DER	WER
With Last Letter	4.35%	12.6%
Without Last Letter	4.33%	12.51%

4.5 Comparison

The proposed hybrid system outperforms decision tree model (as the best of letter based approaches) and the combined n-grams models using backoff (as the best of word based approaches). Table 4.7 below shows a comparison of the results of those 3 models when tested against the same test set (with last letter).

Table 4.7: Comparison of the results of the best 3 algorithms

Model Name	DER	WER
Hybrid System	**4.35%**	**12.6%**
Decision Tree	10.84%	31.6%
Combined N-grams Models Using Backoff	5.64%	12.94%

4.6 Limitations

- The training and testing of the classifiers is done on a relatively small dataset (10 books for letter based approaches and 16 books for word based ones). Increasing the size of the dataset is a trade-off between decreasing the error rate and increasing the amount of resources required to run the algorithms.
- Another limitation is that not all the classifiers have been trained and tested. Increasing the number of classifiers may increase the chance of finding a better classifier that gives better results.

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, the aim was to implement and test various models from two worlds: letter based approaches and word based approaches to automatically restore diacritics for Arabic text. And also build a hybrid model that combines the best two models from both worlds in order to obtain a better one. In letter based approaches, Decision Tree, Naïve Bayes, Support Vector Machines and Multi-layer Perceptron were implemented and tested. In word based approaches, different n-grams models were implemented and tested individually and together using backoff.

Those models were tested and evaluated against Taskeela dataset. The results showed that Decsion Tree was the best in letter based approaches and Combined N-grams Models Using Backoff was the best in word based approaches. The hybrid model that combines those two models outperforms both with results in DER and WER of 4.35% and 12.6%; and 4.33% and 12.51% when ignoring the last letter.

5.2 Future Work

Additional improvements could be made to further decrease the error rates of the various approaches considered by this thesis:

- Increasing the dataset size so as to increase the chances of returning a correct label (diacritic) since the classifier will be more trained and experienced.
- Researching other approaches especially in neural networks field that may handle the problem with better results.
- Adding a model that can predict the syntactic diacritics for a word, i.e., predicting the diacritic of the last letter of the word depending on its role within a given sentence.

Appendix

Appendix

Lists

ML	Machine Learning
NLP	Natural Language Processing
TTS	Text To Speech
POST	Part Of Speech Tagging
WSD	Word Sense Disambiguation
MT	Machine Translation
AI	Artificial Intelligence
RE	Regular Expression
NBC	Naïve Bayes Classifier
SVM	Support Vector Machines
NLTK	Natural Language Toolkit
MLP	Multi-layer Perceptron
LTSM	Long Short Term Memory
RNN	Recurrent Neural Networks
WER	Word Error Rate
DER	Diacritization Error Rate
SER	Segment Error Rate
UTF-8	8 Bits Unicode Transformation Format
OOV	Out Of Vocabulary

List of Figures

List of Tables

Bibliography

[1] How to choose machine learning algorithms | microsoft docs. https://docs.microsoft.com/en-us/azure/machine-learning/machine-learning-algorithm-choice, 3 2017. (Accessed on 04/05/2017).

[2] Gheith A Abandah, Alex Graves, Balkees Al-Shagoor, Alaa Arabiyat, Fuad Jamour, and Majid Al-Taee. Automatic diacritization of arabic text using recurrent neural networks. *International Journal on Document Analysis and Recognition (IJDAR)*, 18(2):183–197, 2015.

[3] Eneko Agirre and Philip Edmonds. *Word sense disambiguation: Algorithms and applications*, volume 33. Springer Science & Business Media, 2007.

[4] Marian Stewart Bartlett, Gwen Littlewort, Mark Frank, Claudia Lainscsek, Ian Fasel, and Javier Movellan. Recognizing facial expression: machine learning and application to spontaneous behavior. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 2, pages 568–573. IEEE, 2005.

[5] Steven Bird, Ewan Klein, and Edward Loper. *Natural language processing with Python: analyzing text with the natural language toolkit.* " O'Reilly Media, Inc.", 2009.

[6] Gobinda G Chowdhury. Natural language processing. *Annual review of information science and technology*, 37(1):51–89, 2003.

[7] Thierry Dutoit. High-quality text-to-speech synthesis: An overview. *Journal of Electrical and Electronics Engineering Australia*, 17:25–36, 1997.

[8] Steve R Gunn et al. Support vector machines for classification and regression. *ISIS technical report*, 14:85–86, 1998.

[9] Simon Haykin and Neural Network. A comprehensive foundation. *Neural Networks*, 2(2004):41, 2004.

[10] James H Martin and Daniel Jurafsky. Speech and language processing. *International Edition*, 710, 2000.

[11] Kevin P Murphy. Naive bayes classifiers. *University of British Columbia*, 2006.

[12] Ani Nenkova and Kathleen McKeown. A survey of text summarization techniques. In *Mining text data*, pages 43–76. Springer, 2012.

[13] Lawrence R Rabiner and Biing-Hwang Juang. Fundamentals of speech recognition. 1993.

[14] Mohsen Rashwan, Mohammad Al-Badrashiny, Mohamed Attia, and Sherif Abdou. A hybrid system for automatic arabic diacritization. In *The 2nd International Conference on Arabic Language Resources and Tools*, 2009.

[15] Stuart Russell, Peter Norvig, and Artificial Intelligence. A modern approach. *Artificial Intelligence. Prentice-Hall, Egnlewood Cliffs*, 25:27, 1995.

[16] Vladimir Naumovich Vapnik and Vlamimir Vapnik. *Statistical learning theory*, volume 1. Wiley New York, 1998.

[17] Lin Wu and Pierre Baldi. A scalable machine learning approach to go. *Advances in Neural Information Processing Systems*, 19:1521, 2007.

[18] Imed Zitouni, Jeffrey S. Sorensen, and Ruhi Sarikaya. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics*, ACL-44, pages 577–584, Stroudsburg, PA, USA, 2006. Association for Computational Linguistics.