Karlsruhe Institute of Technology

Deep Identification of Arabic Dialects

Bachelor's Thesis
by

Alaa Mousa

Department of Informatics
Institute for Anthropomatics and Robotics
Interactive Systems Labs

First Reviewer:	Prof. Dr. Alexander Waibel
Second Reviewer:	Prof. Dr.-Ing. Tamim Asfour
Advisor:	Juan Hussain, M.Sc.

Project Period: 10/11/2020 – 10/03/2021

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe, 01.03.2021

. . . . . .

Alaa Mousa

Abstract

Due to the social media revolution in the last decade, Arabic dialects have begun to appear in written form. The problem of automatically determining the dialect of an Arabic text remains a major challenge for researchers. This thesis investigates many deep learning techniques for the problem of automatically identifying the dialect of a text written in Arabic. We investigate three basic models, namely a recurring neural network (RNN) -based model and an unidirectional, short-term memory (LSTM) -based model, and a bidirectional LSTM-based model combined with a self-attention network. We also explore how applying some techniques like convolution and Word2Vec embedding on the input text can improve the achieved accuracy. Finally, we perform a detailed error analysis that considers some individual errors in order to show the difficulties and challenges involved in processing Arabic texts.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The process of computationally identifying the language of a given text is considered the cornerstone of many important NLP applications such as machine translation, social media analysis, etc. Since the dialects could be considered as a closely related languages, dialect identification could be referred to as a special (more difficult) case of language identification problem. Previously, written Arabic was mainly using a standard form known as Modern Standard Arabic (MSA). MSA is the official language in all countries of Arabic world; it is mainly used in formal and educational contexts, such as news broadcasts, political discourse, and academic events. In the last decade, Arabic dialects have begun to be represented in written forms, not just spoken ones. The emergence of social media and the World Wide Web has played a significant role in this development in the first place due to their interactive interfaces. As a result, the amount of written dialectal Arabic (DA) has increased dramatically. This development has also generated the need for further research, especially in fields like Arabic Dialect Identification (ADI).

The language identification problem has been classified as solved by McNamee in [34]; unfortunately, this is not valid in the case of ADI, due to the high level of morphological, syntactic, lexical, and phonological similarities among these dialects (Habash [23]). Since these dialects are used mainly in unofficial communication on the world wide web, comments on online newsletters, etc., they generally tend to be corrupt, and of lower quality (Diab [10]). Furthermore, in writing online content (comments, blogs, etc.) writers often switch between MSA and one or more other Arabic dialects. All of these factors contribute to the fact that the mission of processing Arabic dialects presents a much greater challenge than working with MSA. Accordingly, we have seen a growing interest from researchers in applying NLP research to these dialects over the past few years.

In this thesis, we perform some classification experiments on Arabic text data contains 4 Arabic varieties: MSA, GLF, LEV and EGY. The goal of these experiments is to automatically identify the variety of each sentence in this data using the best deep neural network techniques proposed in the literature and which proved the best results. For our experiments we use three baseline models which are a word based recurrent neural network RNN, a word based unidirectional long-short term memory LSTM and a bidirectional LSTM with self attention mechanism (biLSTM-SA) introduced in (Lin et al. [31]) which proved its effectiveness for sentiment analysis problem. In order to improve the results and gain more accuracy in our experiments

we add more components for the biLSTM-SA model, which achieved the best results. First, we add two convolutional layers to this model in order to gain more semantics features, especially between the neighboring words in the input sequence, before they passed on to LSTM layer. Where we were inspired with this idea from the work of (Jang et al.[25]). Second, we perform a continuous bag of words CBOW embedding over the training data set to gain a thoughtful word encoding that includes some linguistic information as initialization, instead of random embedding, hoping to improve the classification process. We also test two types of tokenization for our best model, namely, the white space tokenization and sentence piece tokenization.

We use for our experiments the Arabic Online Commentary dataset (AOC) introduced by (Zaidan et al.[57]) which contains 4 Arabic varieties, namely, MSA. EGY, GLF and LEV. We perform three main experiments: The first one is to classify between MSA and dialectal data in general, the second one is to classify between three dialects: EGY, LEV, GLF and the final one is to classify between those 4 mentioned varieties. At the end, we perform a detailed error analysis in order to explain the behaviour of our best model and interpret the challenges of Arabic dialect identification problem, especially for written text, and give some suggestions and recommendations may lead to improve the achieved results in future works.

This thesis is structured as follows: First, we present the background of Arabic dialects, their spread's places, and their speakers. Then, in the next chapter, we talk about the challenges and difficulties of Arabic dialect identification automatically.In the fourth Chapter, we give a brief overview about the ADI approaches used in literature for this problem. In the fifth chapter, we talk about the theoretical backgrounds of neural networks, since it is the focus of our research in this work.Then we review in the next chapter the methodologies used in our experiments in details. Finally, in the last chapter, we present the data set we used in details, then we explain the experimental setups and the results we got for all models. After that in the same chapter, we perform a detailed error analysis for the best model, for each experiment separately and then we discuss the effect of each component we added to our model, on the results and give our recommendations in the conclusion section.

Chapter 2

Language Background

In this chapter, we provide a brief overview about the Arabic language, its varieties, its dialects, and the places where these dialects are distributed. We also talk about the difficulty of these dialects and the other languages that have influenced them over time.

2.1 Arabic Language & Arabic Dialects

Arabic belong to the Semitic language group and it contains more than 12 million vocabularies which make it the most prolific language in vocabulary. Arabic has more than 423 million speakers. In addition to the Arab world, Arabic is also spoken in Ahwaz, some parts of Turkey, Chad, Mali and Eritrea. Arabic language can mainly be categorized into the following classes (Samih [46]):

- Classical Arabic (CA)
- Modern Standard Arabic (MSA)
- Dialectal Arabic (DA)

CA exists only in religious texts, pre-Islamic poetry, and ancient dictionaries. The vocabularies of CA are extremely difficult to comprehend even for Arab linguists. MSA is the modern advancement of CA, so it is based on the origins of CA on all levels: phonemic, morphological, grammatical, and semantic with a limited change and development. MSA is the official language in all Arabic speaking countries. It is mainly used in written form for books, media, and education and spoken form for official speeches, e.g. in news and the media. On the other hand, DA is the means of colloquial communication between the inhabitants of the Arab world. These dialects heavily vary based on the geographic region. For instance, the Levant people are not able to understand the dialect of the north African countries. Among the different dialects of the Arab world, some dialects are better understood than others, such as the Levantine dialect and the Egyptian dialect, due to the leadership of countries like Egypt and Syria in the Arab drama world. The main difference between the Arabic dialects lies in the fact that they have been influenced by the original languages of the countries that they are spread in, for example, the dialect of the Levant has been heavily influenced by the Aramaic language (Bassal [4]). In general, the dialects in

the Arab world can be classified into the following five classes as shown in Figure2.1 (Zaidan et al.[58]):

- Levantine (LEV): It spreads in the Levant and is spoken in Syria, Lebanon, Palestine, Jordan and some parts of Turkey. It is considered one of the easiest dialects for the rest of the Arabs, and what contributed to that is the spread of Syrian drama in the Arab world, especially in the last two decades. It has around 35 million speakers. This dialect is heavily influenced by the original Aramaic language of this region as it constitutes about 30 % of its words.
- Iraqi dialect (IRQ): It is spoken mainly in Iraq and Al-Ahwaz and the eastern part of Syria. The number of speakers in this dialect is up to 29 million.
- The Egyptian dialect (EGY): The most understood dialect by all Arabs due to the prevalence of Egyptian dramas and songs. It is spoken by approximately 100 million people in Egypt.
- The Maghrebi dialect (MGH): It spreads in the region extending from Libya to Morocco and includes all countries of North Africa, Mauritania and parts of Niger and Mali. It is considered as the most difficult dialects for the rest of Arabs, especially this spoken in Morocco, due to the strong influence of Berber and French on it. The number of speakers in this dialect is up to 90 million. Branching from it some extinct dialects like Sicilian and the Andalusian dialect.
- The Gulf Dialect (GLF): It is spoken by all Arab Gulf states: Saudi Arabia, Qatar, UAE, Kuwait and Oman.

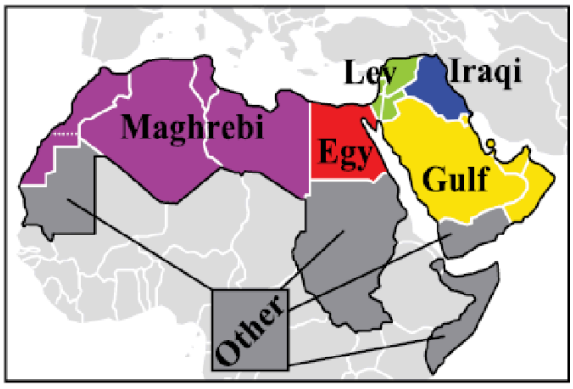

Figure 2.1: Categorization of Arabic dialects in 5 main classes [58]

Chapter 3

Problem of Arabic Dialects Identification

3.1 Definition

The Arabic Dialect Identification (ADI) problem refers to how to automatically identify the dialect in which a given text or sentence is being written. Although the problem of determining the language of a given text has been classified as a solved problem (Mcnamee [34]), the same issue in the case of closely related languages such as Arabic dialects is still considered as a real challenge. This is because, in some particular contexts, this mission is not easy to perform even by humans, mainly because of the many difficulties and issues that exist as we will see in this chapter.

3.2 The Difficulties & Challenges

Besides, they share the same set of letters, and their linguistic characteristics are similar, as we have previously indicated. The following are some difficulties that render the automatic distinction between dialects of Arabic a real challenge for researchers:

- Changing between several Arabic varieties, not only across sentences, but sometimes also within the same sentence. This phenomenon is called "Code-switching" (Samih et al.[48]). It is very common among Arab social media users as they often confuse their local dialect with MSA when commenting (Samih et al.[47]). This increases dramatically the complexity of the corresponding NLP problem(Gamback et al.[18]).
- Due to the lack of DA academies, it suffers from the absence of a standard spelling system (Habash [22]).
- Some dialectal sentences comprise precisely the same words as those in other dialects, which makes it extremely difficult to identify the dialect of those sentences in any systematic way (Zaidan et al.[58]).
- Sometimes the same word has different meanings depending on the dialect used. For example, "tyeb" means "delicious" in LEV and "ok" in EGY dialect (Zaidan et al.[58]).

- Words are now written identically across the different Arabic varieties, but refer to entirely different meanings since short vowels are substituted by diacritical marks instead. The reason for this is that most current texts, (including texts written in MSA) ignore the diacritical marks, and readers are left with the task of inferring them from context. For instance, the word "nby" means in GLF "I want to" and is pronounced in the same way as in MSA and almost all other dialects, but in MSA it means "prophet" and is pronounced as "nabi"(Zaidan et al.[58], Althobaiti [3]).
- Spread the Arabizi writing phenomenon, which involves writing the Arabic text in Latin letters and replacing the letters that do not exist in Latin with some numbers. The Arabizi was created during the new millennium with the appearance of some Internet services which used to support Latin letters as the only alphabet for writing. Which forced many Arabs to use the Latin alphabet. Transliteration of Arabizi does not follow any guidelines or laws, causing confusion and uncertainty, which makes it difficult to recognize Arabic dialects from written texts (Darwish et al.[9]).
- Certain sounds that are absent from the Arabic alphabet have a direct influence on the way certain letters are pronounced in some dialects. Hence, many people tend to use new letters borrowed from the Persian alphabet, for example, to represent some sounds, such as "g" in German and "v,p" in English. These attempts to expand the Arabic alphabet when writing dialectal texts resulted more variations between dialects [3].
- When compared to MSA the number of annotated corpora and tools available for dialectal Arabic is currently severely limited because the majority of the earlier research had focused on MSA [3].
- Some Arabic letters are pronounced differently depending on the dialect used, and this is one of the prime differences between dialects. However, the original letters are used when writing, which makes the sentences look similar and hides the differences between them. For example the letter ق in Arabic is pronounced in three possible ways based on the used dialect: short "a" or "q" or as "g" in German.

3.3 Application of ADI

In this section we present some useful application of ADI as they discussed in (Zaidan et al.[58]):

- The ability to distinguish between DA and MSA is useful for gathering dialectal data that can be used in important applications such as building a dialect speech recognition system.
- By identifying the dialect of the user, apps can tailor search engine results to meet his specific requirements, and also predict which advertisements he is likely to find interesting.
- When a Machine Translation (MT) system could identify the dialect before processing, it attempts to discover the MSA synonyms of not recognized words instead of ignoring them, which improve its performance.

Chapter 4

Arabic Dialect Identification Approaches

Since dialects are considered as a special case of languages as we mentioned earlier, the problem of determining the dialect of an Arabic written text (ADI) is similar to the problem of determining the language of the text, from a scientific point of view. In this chapter, we give a brief historical overview about the most important techniques mentioned in the literature for both ADI and language identification (LI).

4.1 Minimally Supervised Approach

The Methods of this approach were used in early works (a decade ago) for ADI because the available DA datasets were very limited and almost non-existent. The basic work of these methods relies on the word level, that is, to classify each word in the given text and then trying to classify the whole text based on the combination of those classifications. Therefore, these methods depend primarily on dictionaries, rules and morphological analyzers. As an example, we give in the following a brief overview of some of these methods that called lexicon based methods(alshutayri et al.[2]).

4.1.1 Dialectal Terms Method

In this method a dictionary for each dialect will be generated. And then the MSA words will be deleted from the given text by comparing them with the MSA word list. After that, each word will be classified depending on in which dictionary it will be found. Finally, the dialect of the text will be identified depending on those Previous word level classifications. Figure 4.1 [2] illustrates the process of lexicon based methods.

4.1.2 Voting Methods

In this kind of methods, the dialect classification of a given text is handled as a logical constraint satisfaction problem. In the following we will see two different types of voting methods [2]:

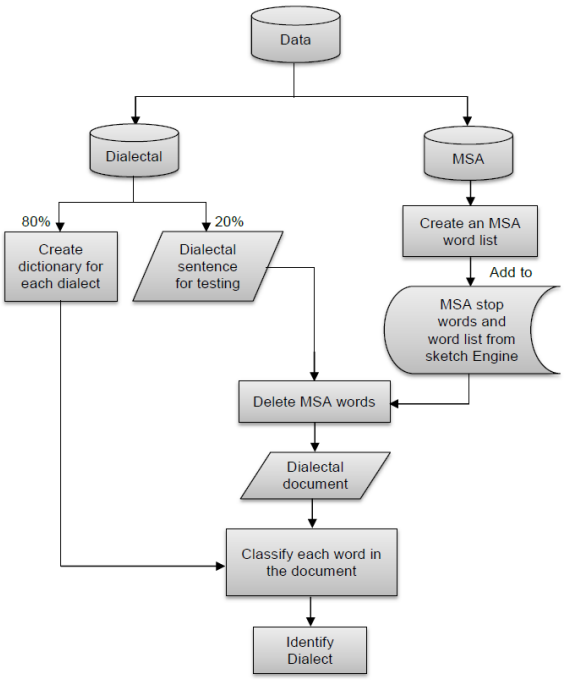

Figure 4.1: Classification process using lexicon based approach [2]

4.1.2.1 Simple Voting Method

In this method, as in dialectal term method, the dialect of each word will be identified separately by searching in the dictionaries of relevant dialects. For the voting process, this method builds a matrix for each text, where each column represents one dialect and each row represents a single word. The entries of this matrix will be specified based on the following equation:

$$a_{ij} = \begin{cases} 1 & \text{if } word\, i \in dialect\, j; \\ 0 & otherwise. \end{cases} \tag{4.1}$$

After that, the score for each dialect will be calculated as the sum of the entries of its own column, and this sum exactly represents the number of words belong to this dialect. Finally, the dialect with the highest score will win.To treat cases in which more than one dialect has the same score, the authors in [2] introduced the weighted voting method described in the next section.

4.1.2.2 Weighted Voting Method

The entries of the matrix in this method will be calculated differently. Instead of entering 1 if the word exists in the dialect, the probability of belonging this word to the dialect will be entered. This probability will calculated as shown in the following equation:

$$a_{ij} = \begin{cases} \frac{1}{m} & \text{if } word\, i \in dialect\, j; \\ 0 & otherwise. \end{cases} \tag{4.2}$$

Where m represents the number of dialects containing the word. This way of calculation gives some kind of weight to each word, therefore it reduces the probability for many dialects to have the same score.

4.1.3 Frequent Terms Methods

The weight of each word in these methods, will be calculated as a fraction of word frequency in the dialect divided by the number of all words in the dictionary of this dialect. Therefore, the dictionary for each dialect contains, besides the words, their frequencies, which were calculated before the dictionary was created. According to [2], considering the frequency of the word improve the achieved accuracy compared to the previous methods. The weight will be calculated as a fraction of word frequency in the dialect, divided by the number of all words in the dictionary of this dialect, as follows:

$$W(word, dict) = \frac{F(word)}{L(dict)} \tag{4.3}$$

F(word) is the frequency of the word in the dialect and L(dict) is the length of dialect dictionary (The number of all words in the dictionary).

4.2 Feature Engineering Supervised Approach

In order to identify the dialect of a particular text, this approach requires relatively complex feature engineering steps to be applied to that text before it is passed to a classifier. These steps represent the given text by a numerical values so that in the next step a classifier can classify this text into a possible class, based on these values or features. For our problem the possible classes are the dialects, with which the text is written.

4.2.1 Features Extraction

In this section we describe two of the most important methods used in the literature for extracting features of a text in order to identify its dialect with one of the traditional machine learning methods will be described in the next section. These methods are the bag of word (CBOW) and the n-gram language modelling method.

4.2.1.1 Bag-of-words model (BOW)

BOW as described in (Mctear et al.[35]) is a very simple technique to represent a given text numerically. This technique considers two things for each word in representation: The appearance of the word in the text and the frequency of this appearance. Accordingly, this method represents a given text T by a vector $v \in \mathrm{R}^n$, where n is the number of words in the given text. Each element x_i in v represents two things as mentioned before: If the word i appears in the text, and how many times. One disadvantage of this method is that it ignores the context of the words in the text (such as the order or the structure). Another problem is that generating a big vectors in case of big sentences increases the complexity of the problem dramatically.

4.2.1.2 n-gram language model

Representing the text by only considering the appearance of its words and its occurrences lead to the loss of some contextual information, as we saw in the last section. To avoid such problems, n-gram approach described in (Cavnar et al.[7]) is used. This approach considers the N consecutive elements in the text instead of single words. Where these elements could be characters or words. The following example

illustrates the idea of character n-gram for the word ”method”:

unigram: m, e, t, h, o, d

bigram: _m, me, et, th, ho, od, d _

trigram: _me, met, eth, tho, hod, od _

And the word n-gram for the sentence ”This is the best method” would be:

unigram: This, is, the, best, method

bigram: This is, is the, the best, best method

trigram: This is the, is the best, the best method

4.2.2 Classification Methods

After extracting the features of a particular text using methods such as those described in the previous section, the traditional machine learning algorithms such as Support Vector Machine (SVM), Decision Rules (LR) and Naive Bayes (NB) described in this section receive these features as input to identify the dialect of this text.

4.2.2.1 Logistic Regression(LR)

LR (Kleinbaum et al.[29]) is a method for both binary and multi class classification. Where for our problem each class represents a possible dialect. The LR model is a linear model predicts an outcome for a binary variable as in the following equation:

$$p(y_i|x_i, w) = \frac{1}{1 + exp(-y_i w^T x_i)} \tag{4.4}$$

Where y_i is the label of example i and x_i is the feature vector of this example. To predict a class label, LR uses an iterative maximum likelihood method. To calculate the maximum likelihood estimation (MLE) of class w, the logarithm of the likelihood function in observed data will be maximized. The following formula represents this likelihood function:

$$\prod_{i=1}^{n} \frac{1}{1 + exp(-y_i w^T x_i)}$$

So, the final formula for the MLE of calss w will be:

$$MLE(w) = argmax_w - \sum_{i=1}^{n} ln(1 + exp(-y_i w^T x_i)) \tag{4.5}$$

4.2.2.2 Support Vector Machine(SVM)

SVM (Cortes and Vapnik [8]) is a machine learning algorithm used widely for language identification tasks. SVM works as follows: it tries to split the data points which represent the features extracted from the text into two classes (where each class represents a possible dialect in the problem) by creating a hyperplane depending on support vectors (SV). SV are the closest data points to the hyperplane and they play the major role in creating it, because the position of this hyperplane will be specified based on those SV. The distance between the hyperplane and any of those SV is called a margin. The idea of SVM is to maximize this margin, which increase the probability that new data points (features) will be classified correctly [26]. Figure 4.2 illustrates how SVM works.

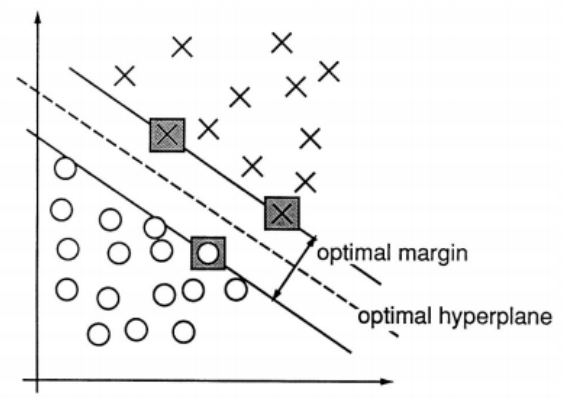

Figure 4.2: SVM classifier identifies the hyperplane in such a way that the distance between the two classes is maximal [8]

4.2.2.3 Naive Bayes(NB)

NB is a simple and powerful classification technique based on Bayes' theorem (Lindley et al.[32]).The NB classifier assumes the stochastic independence between the features of a dialect, although these features could have interdependence between themselves (Mitchell et al.[38]). NB combine both prior probability as well as the likelihood value to calculate the final estimation value called the posterior probability as in the following equation:

$$P(L|features) = \frac{P(feature|L)P(L)}{P(features)} \tag{4.6}$$

Where P(L|features) is the posterior probability, P(feature|L) is the likelihood value, P(L) is the dialect prior probability and P(features) is the predicted prior probability. To illustrate this equation, we consider the following example shown in Figure 4.3 (Uddin [53]).

The NB classifier classifies the white data point as follows [53]: First, it calculates the prior probabilities for the both classes green (G) and red (R). The prior probability of being green is two times that of being red because the number of the green data points is two times the number of the red data points. Accordingly, P(G) = 0.67 (40/60) and P(R) = 0.33 (20/60). Now to calculate the likelihood values P(W|G) (white given green) and P(W|R) (white given red) we draw a circle around the white data point and counts the green and the red data points inside this circle.

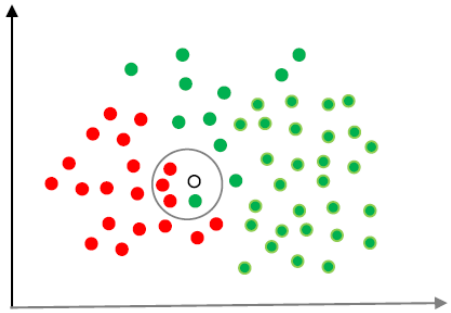

Figure 4.3: Classify the new white data point with NB classifier [53]

We have one green and 3 red data points, so, P(W|G) = 1/40 and P(W|R) = 3/20. Finally, by applying the equation (5.6) to calculate the posterior probabilities P(G|w) = P(G) * P(W|G) = 0.017 P(R|W) = P(R) * P(W|R) = 0.049. So, this white data point will be classified as red.

4.3 Deep Supervised Approach

Although the traditional machine learning algorithms such as LR, SVM and NB...etc, have proved their effectiveness in many AI problems, they have a limitation prevent them to act perfect in some real world problems. The following points presents some of these limitations [46]:

- Their simple structures limit their ability to represent some information about real world problems.
- These linear models are often unable to explore non-linear dependencies between the input and the features.
- These methods often based on features which very hard to extract.
- Extracting the features and training in these methods are going on separately, which prevent the overall optimization of the performance.

Such limitations caused that many AI researchers moved to more complex non-linear models such as deep neural networks DNN introduced in chapter 5. Recently, DNN has proven its superiority over many traditional machine learning techniques in many fields (kim [28]). In this section we did not present the techniques of DNN because we specify chapter 5 for them, as they are the main techniques we used for our experiments.

4.4 Related Work

In this section, we give an overview about the most related researches to the topic of Arabic dialect language identification systems.

Zaidan and Callison-Burch research in their two works [57][58], the use of language modelling (LM) methods, with different n-gram (1,3,5) on character and word level as future extraction methods, for ADI. In the work of [57] they examined a word

trigram model for Levantine, Gulf, Egyptian and MSA Sentences. Their model results was as follows: 83.3% accuracy for classification between the dialects (Levantine vs Gulf vs Egyptian) and 77.8% accuracy in the case of (Egyptian vs MSA) and only 69.4% for the 4-way classification. On the other hand, in [58], they trained word 1-gram, 2-grams, and 3-grams models, and character 1-graph, 3-graph, 5-graph models, on an Arabic online-commentary (AOC) dataset. The best results obtained by 1-gram word-based and 5-graph character-based models, for 2-way classification (MSA vs dialects), with 85.7% and 85.0% accuracy, respectively.

Elfardy and Diab [15], introduced a sentence-level supervised approach which can classify MSA and EGY. They used the WEKA toolkit introduced in Smith and Frank [51], to train their Naive-Bayes classifier, which achieved 85.5% classification accuracy on AOC dataset.

In Elfardy et al.[13], the authors adapted their system proposed earlier in Elfardy et al.[14], which identify linguistic code switching between MSA and Egyptian dialect. The system based on both morphological analyzer and language modeling and tries to assign words in a given Arabic sentence to the corresponding morphological tags. For that they used the word 5-grams. To train and test their model, the authors created an annotated dataset with morphological tags. The language model was built using SLIRM toolkit introduced in Stolcke[52]. To improve the performance of this system, a morphological analyzer presented in Pasha et al.[39] MADAMIRA, was used. This new adaption reduced the analyses complexity for the words, and enabled the adapted system AIDA to achieve 87.7% accuracy for the task of classification between MSA and EGY.

Malmasi et al.[33], presented a supervised classification approach for identifying six Arabic varieties in written form. These varieties are MSA, EGY, Syrian(SYR), Tunisian (TUN), Jordanian (JOR) and Palestinian (PAL). Both AOC dataset and the "Multi-dialectal Parallel Corpus of Arabic" (MPCA) released by Habash et al. [21], were used for training and testing. Authors employed Character n-gram as well as Word n-gram for feature extraction, and LIBLINEAR Support Vector Machine(SVM) package introduced in [17], for classification. The work achieved best classification accuracy of 74.35%.

The first work tried to classify the dialect on city level was the work of Salameh et al.[45]. The authors built an Multinomial Naive Bayes(MNB) calssifier to classify the dialects of 25 various cities in Arab world, some of them are located in the same country. They used the MADAR corpus, presented in Bouamor et al.[6], which contains besides those 25 dialects, sentences in MSA, English and French. To train the MNB model, a combination of character n-gram(1,2,3) with word unigram was used. To improve the training process they built both character and word 5-gram model for each class in corpus, where the scores of these models were used as extra features. The results obtained with their model was as following, 67.5% accuracy on MADAR corpus-26 and 93.6% accuracy on MADAR corpus-6. Authors also examined the effect of sentence length on the classification accuracy achieved, and found that sentences with an average length of 7 words, have been classified with only 67.9% accuracy versus more than 90% for those with 16 words.

The work of Eldesouki et al.[12] examined several combination of futures with several classifiers such as MNB, SVM, neural networks and logistic regression. The goal was

to build a 5-way classifier (EGY vs LEV vs MAG vs MSA). The best accuracy of 70.07% were achieved with SVM trained on character (2,3,4,5)-gram.

Elaraby and AbdulMajeed [11] performed several experiments based on LR classifier for ADI task on the AOC data set. The task was to classify 4 Arabic varieties (LEV, EGY, GLF, MSA). The authors used two type of feature representation technique (presence vs. absence) as well as TF-IDF (Robertson et al.[42]) to represent the word (1-3)-gram features. The results were as following: The classifier achieved accuracy of 83.71% with (presence vs. absence) and 83.24% with TF-IDF in binary classification experiment (MSA vs.dialectal Arabic). An accuracy of 78.24% in 4-way classification experiment (LEV, EGY, GLF, MSA) for the two mentioned type of feature representation.

Sadat et al.[44] performed two sets of experiments to classify 18 various Arabic dialects. Moreover, training and testing data were collected from the social media blogs and forums of 18 Arab countries. Authors tested three character n-gram features, namely (1-gram, 2-grams and 3-grams), first for experiment of Markov language model and then for experiment of Naive Bayes classifier. The best accuracy (98%) achieved by the Naive Bayes classifier trained with character bi-gram.

Guggilla [20] presented a deep learning system for ADI. The architecture of the system based on CNN and consists of 4 layers. The first layer calculates the word embedding for each word in the input sentence randomly in the range [-0.25,0.25]. This Embedding layer is followed by a convolutional, max pooling and a fully connected softmax layer, respectively. Das system achieved 43.77% classification accuracy for distinguishing between EGY, GLF, MSA, MAG, LEV.

Ali [1] examined a CNN architecture works on character level to classify 5 Arabic dialects (GLF, MSA, EGY, MAG,LEV). This architecture consists of 5 sequential layers. The input layer maps each character in the input into a vector. The rest 4 layers are as following: Convolutional layer, max pooling layer and 2 sequential fully connected softmax layers. System achieved classification accuracy of 92.62% .

In the work of Abdul Majeed et al.[11] authors performed several experiments on AOC data set to distinguish between MSA, DIAL (2-way) and GLF, LEV, EGY (3-way) and finally between the 4 varieties MSA, GLF, EGY, LEV (4-way). The best accuracy achieved were with Bidirectional Gated Recurrent Units (BiGRU) with pre-trained word embedding for the 2-way experiment (87.23%) and with NB classifier (1+2+3 grams) for the 3-way experiment (87.81%) and with an Attention BiLSTM model (with pre-trained word embedding) for the 4-way experiment (82.45%).

Table 4.1 gives a summarizing of these works and contains the most important information such as used model, features, data set and the achieved accuracy.

Reference	Model	Features	Dialects	Corpus	Acc
Zaidan et.al[57]	LM	W 3-grams	L-G- E	AOC	83.3
Zaidan et.al[57]	LM	W 3-grams	M-E	AOC	77.8
Zaidan et.al[57]	LM	W 3-grams	M-E-G-L	AOC	77.8
Zaidan et.al[58]	LM	W 1-gram	M-D	AOC	85.7
Zaidan et.al[58]	LM	Ch 5-graph	M-D	AOC	85.0
Elfardy et.al[15]	NB	WEKA	M-E	AOC	85.5
Elfardy et.al[13]	LM	W 5-grams	M-E	AOC	87.7
Malmasi et.al[33]	SVM	Ch 3-grams	6 country-level	AOC	65.26
Malmasi et.al[33]	SVM	Ch 1-4grams	6 country-level	AOC	74.35
Salameh et.al[45]	MNB	Ch 1-3grams	25 city-level	MADAR26	67.5
Salameh et.al[45]	MNB	Ch 1-3grams	25 city-level	MADAR6	67.5
Desouki et.al[12]	SVM	Ch 2-5grams	E-L-N-M-G	DSL2016	70.07
Elaraby et.al[11]	LR	W1-3grams	L-E-G-M	AOC	78.24
Elaraby et.al[11]	LR	W 1-3grams	M-D	AOC	83.24
Sadat et.al[44]	NB	Ch 1-3grams	18country-level	Own corpus	98.0
Guggilla [20]	CNN	W embedding	E-G-N-L-M	DSL2016	43.77
Ali [1]	CNN	Ch embedding	E-G-N-L-M	DSL2018	92.62
Elaraby et.al[11]	BiGRU	W embedding	M-D	AOC	87.23
Elaraby et.al[11]	NB	W 1-3grams	E-G-L	AOC	87.81
Elaraby et.al[11]	BiLSTM	W embedding	M-G-L-E	AOC	82.45

Table 4.1: summarizing of related works, where in feature column W denote to word and Ch denote to character. The varieties MSA, GLF , LEV, EGY, DIAL represented with M, G, L ,E ,D, respectively and North African dialect with N, for simplicity.

Chapter 5

Deep Neural Networks

We have seen in the previous chapter how is useful to use approaches like Deep Neural Networks (DNN) to handle some real world problems related to artificial intelligence such as NLP. We have also seen some of their advantages over the traditional future engineering supervised approaches and the increasing interest of artificial intelligence researchers in using it. We specify this chapter to present the idea behind DNN, its types and to give a brief overview about how it works. This chapter serves as a basis to understand the methodology we used later in all out experiments.

5.1 Basics of DNN

As we know, human brain uses a huge amount of connected biological neurons to process the external stimulation(inputs) and make decisions based on previous knowledge. The idea behind the DNN is to imitate this biological methodology hoping to gain its high performance.

5.1.1 The Artificial Neuron

The artificial neuron is a processing unit represents a simple abstraction of the biological neuron. The structure of it illustrated in figure 5.1 [46].

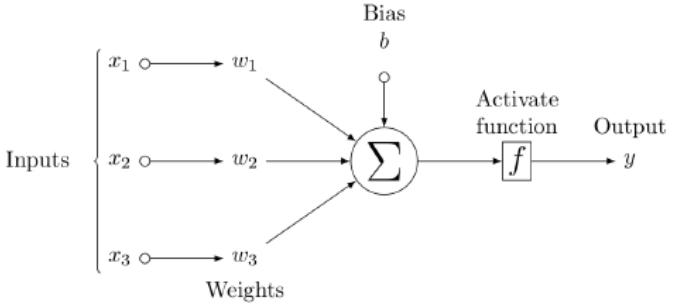

Figure 5.1: The artificial neuron [46]

Where the input for this neuron is a vector $X \in \mathrm{R}^3$ consists of the features (x_1,x_2,x_3). The linear combination of vector X with the weights vector W (w_1,w_2,w_3) will be

calculated, where b denotes a bias b and then a nonlinear function f will be applied on the result of this combination to calculate the output as in equation 5.1.

$$y = f(\sum_{n=1}^{n} x_i w_i + b) \tag{5.1}$$

There are many kinds of nonlinear activation functions, but the most used ones are the sigmoid function (called also the logistic function) and has the following mathematical notation:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{5.2}$$

This function maps each input value whatever it is to a value between 0 and 1. The other one is the tangent function which maps each input to a value between -1 and 1 as in the equation:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{5.3}$$

Figures 5.2 and 5.3 show the graphical representation of these two functions, respectively.

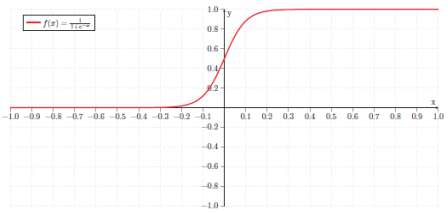

Figure 5.2: Sigmoid function [46]

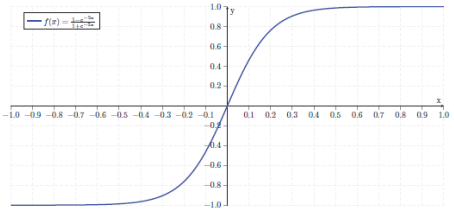

Figure 5.3: Hyperbolic tangent function [46]

5.1.2 General Architecture of DNN

Often a neural network consists of several successive layers. Each layer contains a certain number of neurons described in the previous section. The neurons in each layer are connected to those in the next layer, and so on, respectively, to form the complete multi layer network (rumelhart et al.[43]). The inter-layers are called the hidden layers, while the first and last layers represent the input and output layers, respectively. Figure 5.4 illustrates the idea of multi layer DNN graphically. The distribution of neurons between layers and their connectivity in this way means that the process followed by each neuron will be apply repeatedly on the input. So, we have here a consecutive series of linear combinations which means a series of weights matrix multiplications, with an activation function for each.

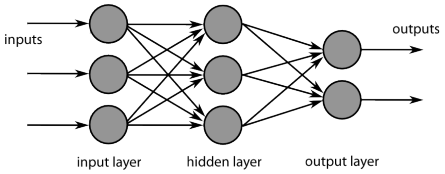

Figure 5.4: Multi-Layer Neural Network [56]

5.2 Types of DNN

In this section we explain the different types of DNN and their most important uses.

5.2.1 Convolutional Neural Networks(CNN)

The concept of time delay neural networks (TDNN) introduced by (Waibel et al. [54]) is considered the precursor to the CNNs. TDNN is a feed-forward network was basically introduced for the speech recognition. It has proved its efficiency in handling a long temporal context information of a speech signal by its precise hierarchical architecture, outperforming the standard DNN (Peddinti et al.[40]). Attempts to customize TDNN for image processing tasks led to the CNNs. CNNs as described in (Ketkar[27]) handle the input as matrix with two dimensions. This property made this kind of DNN a perfect choice to process images where each image is considered as matrix its entries are the pixels of this image. Basically, each CNN consists of the following layers [27]:

- Convolutional layer
- Pooling layer
- Fully Connected Layer

The convolutional layer makes some kind of scanning over the input matrix to analyze it and extract some features from it. In other words, some kind of window sliding over the matrix where in each step a filter will be applied on this window (this smaller part of input matrix). This process requires two parameters, the kernel size (e.g 3*3) which denote to the size of this window or the size of this part from the input will be filtered in each step. The second parameter is a step size which denotes to the sliding size in each step. One could imagine this process as a filter with two "small" dimensions moved over a big matrix, with a fixed sliding size, from left to right and from top to bottom. The kind of so called *Padding* determines how this filter should behave on the edges of the matrix. The filter has fixed weights that are used together with the input matrix values in the current window to calculate the result matrix. The size of this result matrix depending on the step size, the padding way, and the kernel size. Usually contains the CNN two sequential convolutional layers, each with 16 or 32 filter. The second one receive the result matrix of the first layer as input. Then these two layers are followed with a pooling layer.

According to [27] the main role of pooling layer is to pass on only the most relevant signal from the result matrix of the convolutional layers. So, it performs a kind of aggregation over this matrix. For example the max pooling layer considers only the

highest value in kernel matrix and ignores all other values. Pooling layer helps in reducing the complexity in the network by performing an abstract representation of the content.

Each neuron in the fully connected layer is connected to all input neurons and all output neurons. This layer is connected to the output layer which has a number of neurons equal the number of classes in the classification problem. The result layer of convolutional and pooling layers must be flattened before it passed on to the fully connected layer. Which means this layer receives the object features without any position information (location independent).

The structure of CNN usually contains two sequential similar convolutional layers followed by a pooling layer, after that again, two convolutional layers followed by a pooling layer. Finally a fully connected layer, followed by the output layer. Figure 5.5 illustrates this structure.

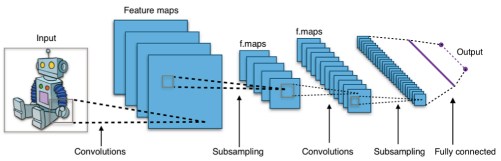

Figure 5.5: Typical Convolutional Neural Network [55]

5.2.2 Recurrent Neural Networks(RNN)

The RNN is a special type of DNN has a recurrent neurons instead of normal feed forward neurons. Recurrent Neuron has unlike normal neuron an extra connection from its output to its input again. This feed back connection enables applying the activation function repeatedly in a loop. In other words, in each repetition the activation learns something about the input and be tuned accordingly. So, over the time these activation represents some kind of memory contains information about the previous input (Elman [16]). This property made the RNN perfect to deal with sequential input data need to be processed in order such as sentences. Thus, RNN were be considered the best choice for NLP problems. Figure 5.6 gives a graphical representation of the recurrent neuron and RNN.

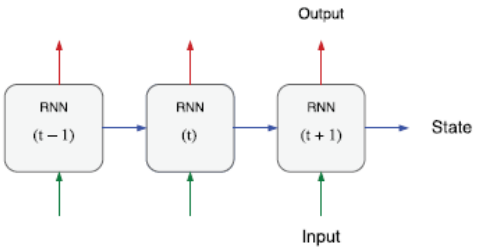

Figure 5.6: Recurrent Neural Network [46]

Where $X = x_1,...,x_n$ is the input sequence and $H = h_1,...,h_n$ represents the hidden states vector. But, RNN has a big problem called the vanishing gradient problem discussed in (bengio et al.[5]). This problem denotes to a drawback during the back

propagation training process over the time. It means that RNN either needs too long time to learn how to store information over the time, or it fails entirely. (Hochreiter and Schmidhuber [24]) solved this problem with their proposed Long short-term memory (LSTM) architecture. LSTM overcome the training problems of RNN by replacing the traditional hidden states of RNN with special memory cells shown in figure 5.7. This memory cells unlike RNN could store information over the time and enabled thereby extracting more contextual feature in the input data (Graves [19]). According to [19], the following composite function calculates the output of LSTM hidden layer:

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + W_{ci}c_{t-1} + b_i) \tag{5.4}$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + W_{cf}c_{t-1} + b_f) \tag{5.5}$$

$$c_t = f_t c_{t-1} + i_t tanh(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \tag{5.6}$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + W_{co}c_t + b_o) \tag{5.7}$$

$$h_t = o_t tanh(c_t) \tag{5.8}$$

Where σ is the sigmoid activation function, i the input gate, f is the forget gate and o is the output gate. (for more details please see [19])

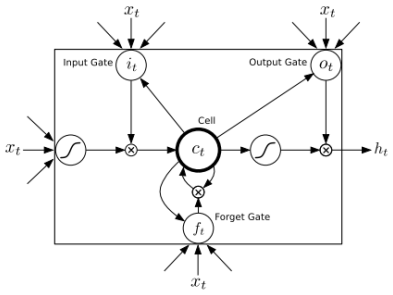

Figure 5.7: Long Short-term Memory Cell [19]

As improvement of LSTM Networks is the bidirectional LSTM (BiLSTM). The idea of bidirectional RNN introduced by (Schuster and Paliwal [49]) was to invest the future context besides the past context to improve the contextual futures extraction process. In other words, the learning process in BiLSTM is performed not only from beginning to the end of the sequential input but also in the opposite direction. So, we have in this architecture two LSTM, one in each direction as shown in figure 5.8.

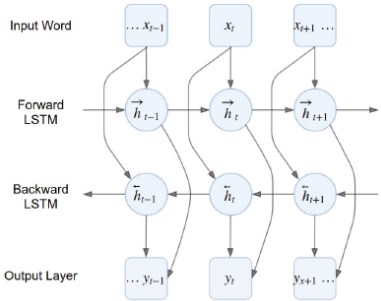

Figure 5.8: Bidirectional LSTM Architecture [46]

5.3 Word Embedding Techniques

Word embedding is the process of mapping phrases or vocabularies into numerical vectors. This mapping can be done randomly or with methods like Glove (Pennington et al.[41]) and Word2Vec (Mikolov et al. [36]), where the distance between these vectors in this case represents information about the linguistic similarity of words. So, we talk here about a special information for each language. In this section we give a brief overview about the two models of Word2Vec embedding method which are continuous bag of word (CBOW) and Skip-gram [36]:

5.3.1 skip-gram

This model tries to learn the numerical representation of some words called target words which are suitable to predict their neighboring words. The number of neighboring words should be predicted is a variable and called a window size, but usually is 4 (Mikolov et al. [37]). Figure 5.9 (right) illustrates the architecture of this model. Where $W_{(t)}$ is a target word and $W_{(t-2)}$, $W_{(t-1)}$, $W_{(t+1)}$, $W_{(t+2)}$ are the predicted context words (assuming that the window size is 4).

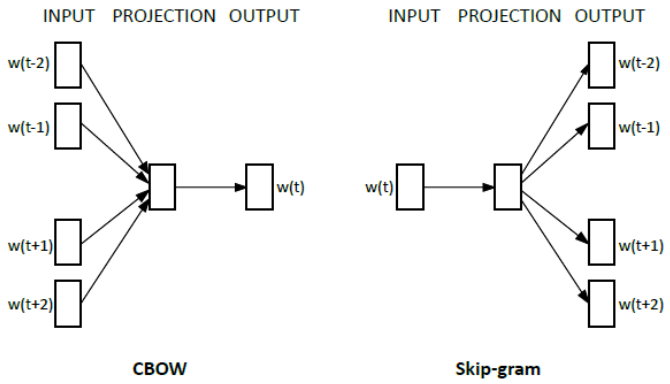

Figure 5.9: The architecture of CBOW and Skip-gram models [36]

5.3.2 continuous bag of words (CBOW)

This model work exactly in the opposite way, it predicts one word from a given context, instead of predicting the context words from a target word as in Skip-gram. The context here could also be one word. This model is visualized in figure 5.9 (left).

We saw in the previous chapter how techniques like BOW only take into account the occurrence of words in the text and their frequency, and completely ignore the context information. In the case of CBOW and Skip-Gram, context is taken into account when the words are represented, which means that closely related words that usually appear in the same context would have a similar representation (similar vectors). For example the names of animals or the names of countries or words like ”king” and ”queen”.

Chapter 6

Methodology

In this section we describe the model architecture that we used in our experiments. We started with a very simple RNN based model. We then moved to an LSTM-based model in order to take the advantages of the LSTM over RNN described in the previous chapter. As the attention mechanism could be very useful to improve the achieved accuracy of a classification problem we then upgraded our model to an bidirectional LSTM with self attention mechanism. We also introduced two hybrid models we performed by adding some extra layers to these three baseline models in order to achieve more classification accuracy. In the following we gave each model a short name for simplicity.

6.1 Word-based recurrent neural Network (RNN):

This is simple model consists of the following layers:

- *Input layer*: This is simply an embedding layer used to map every word in the input sentence to a numeric vector with random values. The dimension of this vector will be tuned for each experiment separately depending on the best achieved accuracy as we will see later.
- *RNN layer*: A bidirectional RNN architecture consisting of two hidden layers. The number of units for each layer (hidden size) will be also tuned experimentally.
- *Output layer*: Which is a linear function simply calculates the likelihood of each possible class in the problem from the hidden units values of the previous layer.

The mathematical structure for this model can be described as follows:

$$h(t) = f_H(W_{GH}x(t) + W_{HH}h(t-1)) \tag{6.1}$$

$$y(t) = f_S(W_{HS}h(t)) \tag{6.2}$$

Where W_{GH}, W_{HH},W_{HS} are the weights between layers. $x(t)$, $y(t)$ represent the input and output vector, respectively. f_H represents the activation function of hidden layers and f_S the activation function of output layer.

6.2 Word-based Long-Short Term Memory (LSTM):

- *Input layer*: The same as the input layer for RNN
- *LSTM layer*: An unidirectional LSTM architecture consisting of number of hidden states as introduced the previous chapter 5. The number of these states (hidden size) will be also tuned experimentally as for RNN.
- *Output layer*: The same as the output layer for RNN.

The mathematical structure for LSTM model is described in section 5.2.2

6.3 Bidirectional LSTM with self attention mechanism (biLSTM-SA):

This model introduced by (Lin et al.[31]) consists of two main components, namely biLSTM and a self-attention network. The first part (biLSTM) receives the embedding matrix which contains the word-based numerical representation of the input sentence and produce a hidden states matrix H, as we will see later. In order to calculate the linear combination between the vectors of H, a self-attention mechanism would be applied many times on H considering different parts of the input sentence. The structure of this model illustrated in Figure 6.1.

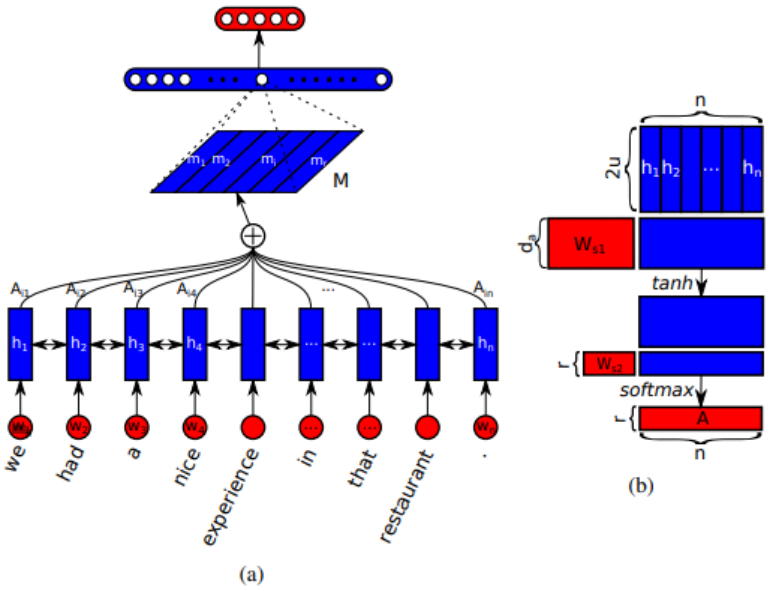

Figure 6.1: biLSTM with self-attention mechanism [31]

The following scenario clarifies the workflow of Figure 6.1 as described in [31]. Assume the input for the proposed model is a sentence consists of n words. First those words will be mapped to a real value vectors (for our experiments we used random embedding). So the output of this step will be a matrix represents the sentence S its columns are those embedding vectors which mean S will have the size n*d. Where d is the embedding size (The dimension of the embedding vectors).

$$S = (W_1, W_2, ...W_n) \tag{6.3}$$

Matrix S will be passed to a biLSTM represented in the following equation which will reproduce and reshape S extracting some relationships between its columns(between the words in the input sentence):

$$\overrightarrow{h_t} = \overrightarrow{LSTM}(w_t, \overrightarrow{h_{t-1}}) \tag{6.4}$$

$$\overleftarrow{h_t} = \overleftarrow{LSTM}(w_t, \overleftarrow{h_{t-1}}) \tag{6.5}$$

So, the produced hidden state matrix H has the size of n*$2u$ where u is the hidden size of the $LSTM$ in one direction.

$$H = (h_1, h_2, ...h_n) \tag{6.6}$$

Now, to calculate the linear combination over the hidden state matrix H, the self-attention in 6.7 will be applied on it. Where w_{s2} is a weight vector of size d_a and W_{s1} is a weight matrix of size d_a*$2u$ and d_a is a hyper parameter. The output a is a vector of values focus, according to [31] on one part of the input sentence. But, the main goal is to obtain the attention over all parts of the sentence. So, the vector w_{s2} in equation 6.7 is replaced by a matrix W_{s2} of size r*d_a where r is also a hyper parameter. Accordingly, the output will be a matrix A represents the applied attention on r parts of the input sentence as in equation 6.8.

$$a = softmax(w_{s2}tanh(W_{s1}H^T)) \tag{6.7}$$

$$A = softmax(W_{s2}tanh(W_{s1}H^T)) \tag{6.8}$$

After that the sentence embedding matrix M will be produced by multiplying the hidden state matrix H with the weights matrix A as follows:

$$M = AH \tag{6.9}$$

Finally, matrix M will be passed through a fully connected and output layer respectively, to eliminate the redundancy problem.

6.4 Hybrid model (CNN-biLSTM-SA):

The model proposed in (Jang et al.[25]) for sentiment analysis problem inspired us to add more component to the previous model (biLSTM-SA) in order to achieve better accuracy. Authors in [25] proved that applying a convolution over the input vectors lead to less dimensions in extracted features before it passed to LSTM which helps it improve its performance. This reduction in dimensions is achieved because this convolution according to [25] can extract some features from the adjacent words in the sentence before it comes to the turn of LSTM to extract the long-short dependencies. Accordingly, we added two $2D$ convolutional layers to the biLSTM-SA model hoping to improve its performance. Each of those layers has a kernel size of (3,3) and a stride of (2,2). Figure 6.2 gives a simplified illustration for the structure of this model.

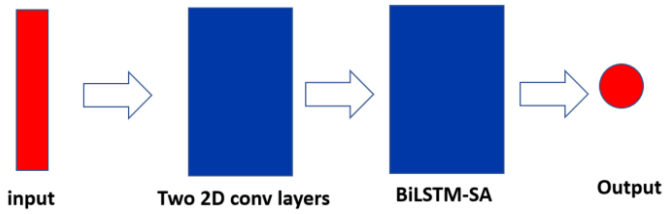

Figure 6.2: CNN-biLSTM-SA model

6.5 Hybrid model (Word2Vec-biLSTM-SA):

Another attempt to improve the performance of biLSTM-SA model was that we added an embedding layer which map the tokens in input to numerical vectors with CBOW word embedding technique introduced in section 5.3. Using this technique instead of generating random values according to [25] lead to a significant improvement in text classification task. For that we used gensim library to calculate this embedding over our training data set where we calculated this embedding with 300 dimensional vectors. Figure 6.3 gives a simplified illustration of this model.

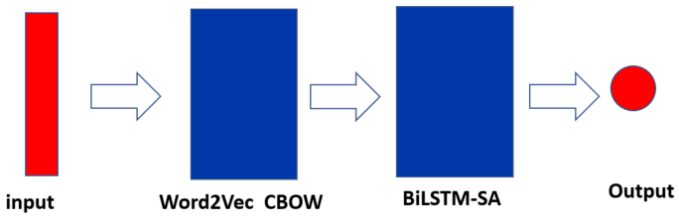

Figure 6.3: Word2Vec-biLSTM-SA model

6.6 Data prepossessing

Since this data were extracted from an online contents, it may be noisy uncontrolled, may have some words in English or in Arabizi (See 3.2). To guarantee that such issues do not affect the performance of our model we performed some prepossessing steps on this data, namely:

- Tokenization: In order to analyse a written text computationally, it should be split into small units called tokens and that is exactly what the tokenization process does. These tokens could be words or even parts of sentence...etc. We used for our experiments the white space tokenization, which means we split the sentences in our dataset into individual words. After we got the best results, as an attempt to achieve more accuracy we used the sentence piece tokenization introduced by (Kudo et al.[30]). This tokenization considers the input as a sequence of Unicode characters independently from the language and apply the byte-pair encoding algorithm (BPE) (Sennrich et al. [50]) on it where the most frequent byte pair considered as one token and the number of unique tokens is predetermined.

- Normalization: As we mentioned earlier, it is possible to write Arabic text with or without diacritics(In this case the diacritics could be extracted from

the context). There are also some letters in Arabic such $Alef Maksura$; which is very commonly to be replaced by normal $Alef$. So, we normalized our data as follows: We cleaned it from any English letters, emoticons, underscore and any letter repetition more than twice. We replaced each $taa´marbuta$ with haa and each $Alefmaksura$ with normal $Alef$. We also replaced all kinds of diacritics $Fatha, Damma, kasra$...etc with nothing.

- Sentence Padding: Padding is the process of making all sentences have the same length. We set this length to 40 in our experiments, which means all sentences longer than 40 words were truncated to be 40, and all sentences shorter than 40 were padded with zero.
- Input quantization: For the white space tokenization we considered only the most repeated 50k words in all our experiments.

Chapter 7

Evaluation

7.1 Data set

We used for our experiments the Arabic Online Commentary dataset (AOC) introduced by (Zaidan and Burch [57]). Zidane and Burch were among the first who research in ADI problem. So, they recognized very early the need of Arabic dialectal data in written form and constructed the AOC dataset. The data in AOC were collected from the comments of users on 3 Arabic online newspapers, the Jordanian *Alghad*, the Egyptian *Alyomalsabea* and the Saudi $Al - Riyadh$ newspapers. Accordingly, it covers 3 Arabic dialects, namely LEV, EGY, and GLF in addition to MSA. AOC contains about 3.1M sentences, only 108.173 of them are labeled using crowdsourcing. We used these AOC labeled data as introduced in the work of (Abdulmajeed et al.[11]), where it was split into 80% for training and 10% for development (validation) and 10% for testing. Table 7.1 gives an overview about the distribution of classes in the dataset.

Variety	MSA	EGY	GLF	LEV	ALL
Train	50,845	10,022	16,593	9,081	86,541
Dev	6,357	1,253	2,075	1,136	10,821
Test	6,353	1,252	2,073	1,133	10,812

Table 7.1: The number of sentences for each Arabic variety in dataset [11]

To illustrate how far these varieties have in common, figure 7.1 [11] display a heat map about the shared vocabularies between each variety and another. As this heat map shows, all mentioned dialects differ from each other more than from MSA, which make the task of distinguishing between these dialects easier than distinguishing them or one of them from MSA. We also can read that both LEV and GLF have significantly more vocabularies in common with the MSA than EGY. That lead, of course, to more difficulties in distinguishing between LEV or GLF and MSA than between EGY and MSA as we will see later in the error analysis of our experiments.

Table 7.1 shows how the number of MSA sentences is many times greater than the number of sentences for each dialect. We will study the effect of that on our results, especially when we try to classify these four varieties from each other in the error analysis section.

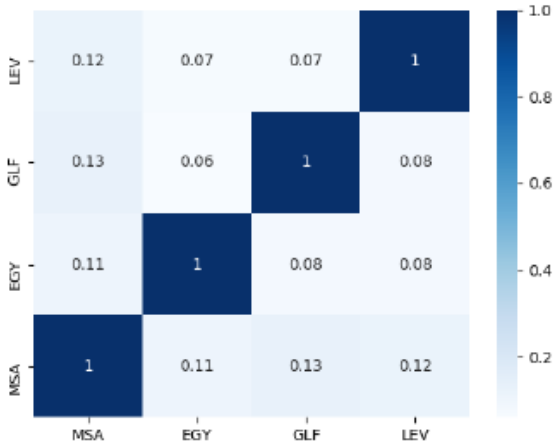

Figure 7.1: Heat map illustrates the percentage of shared vocabularies between varieties in dataset [11]. Please note that this matrix is not symmetric and can be read for example as follows: The percentage of EGY words in GLF is 0.08, whereas the percentage of GLF words in EGY is 0.06.

7.2 Experiments and Results

In this section we explain the experiments we performed and present the results for each experiment in the form of the best achieved classification accuracy by each run for both validation and test data set. We performed three main experiments, where for each experiment we examine the performance of each of our models introduced in chapter 6. The first experiment we called 2-way-experiment, where the task for each model was to classify between MSA and dialectal sentences (DIAL) (two classes) of our data set introduced in the previous section. So, the training, development and test data for the class DIAL in this experiment are the sum of train, development and test data for all 3 dialects introduced in table 7.1. The second experiment we called 3-way-experiment was to classify only between the three dialects GLF, LEV, EGY. Finally, the third experiment we called 4-way-experiment was to classify between the 4 Arabic varieties shown in table 7.1, namely, MSA, GLF, EGY and LEV. All runs shared the same values for some hyper parameters over all experiments. We chose those values by running all our models for each experiment and for each possible value shown in table 7.2 at least 10 times, and registered the values with which we got the best test accuracy.

The rest hyper parameters not mentioned in table 7.2 were chosen for each run separately. We ran for each model nested loops contain too many possible values for each parameter to get the perfect combination of values with which we get the best accuracy. First, we ran two nested loops for embedding dimension and number of LSTM hidden states and we chose the best combination of those parameters. After that we ran another loops to choose the best combination of number of LSTM layers and drop out value and the best number of training epochs.

Parameter	*Best Value*	*Tested Values*
Batch size	16	(8,16,32,64)
Learning rate	0.001	(0.01,0.001,0.0001)
Vocab size	50k	(25k,50k,70k,100k)
Padding value	40	(30,40,50,60)
Optimizer	RMS-prop	(Adam, RMSprop, SGD)

Table 7.2: Shared Hyper parameters over all experiments

Tables 7.3, 7.4 and 7.5 shows the results we got with our models for 2-way, 3-way and 4-way experiment, respectively.

Model	*DEV*	*TEST*
RNN	84.05	84.61
uniLSTM	84.34	85.01
biLSTM-SA	**85.06**	**85.07**
CNN-biLSTM-SA	84.6	84.6
Word2Vec-biLSTM-SA	84.32	84.43

Table 7.3: Achieved classification accuracy for 2-way-Experiment

Model	*DEV*	*TEST*
RNN	83.05	83.11
uniLSTM	83.94	83.51
biLSTM-SA	**85.51**	**83.75**
CNN-biLSTM-SA	85.32	83.72
Word2Vec-biLSTM-SA	84.92	83.19

Table 7.4: Achieved classification accuracy for 3-way-Experiment

As shown in these tables, the proposed model biLSTM-SA outperformed all other models for all experiments we performed. One can also see how the achieved accuracy by all models significantly decreased in the 4-way experiment comparing to other experiments. This denote that the 4-way experiment was the most difficult one for all models. We will see in the following section the possible reasons behind that.

RNN achieved the worse results over all experiment, but with a very little difference from uniLSTM model. One can also read that the difference in results between RNN and uniLSTM as well as between uniLSTM and biLSTM-SA is very low in case of 2-way and 3-way experiment. This is not the case for 4-way-experiment where this difference approaches 1 %, which means that the self attention mechanism affects perfect in more difficult situations.

Unfortunately the results also show that the convolutional layers we added to the biLSTM-SA model to perform CNN-biLSTM-SA model did not help us gaining better results. This model achieved in 2-way and 4-way experiments worse results while it did not lead to any improvement and achieved almost the same accuracy as biLSTM-SA model in the 3-way experiment.

Model	*DEV*	*TEST*
RNN	75.01	76.62
uniLSTM	75.53	77.59
biLSTM-SA	**76.52**	**78.39**
CNN-biLSTM-SA	74.77	76.21
Word2Vec-biLSTM-SA	76.51	76.07

Table 7.5: Achieved classification accuracy for 4-way-Experiment

Results also show that the Word2Vec CBOW embedding layer (the CBOW applied on our training data set) we added to the biLSTM-SA model hoping to gain a thoughtful numerical representation for our vocabularies did not work well and led to lower accuracy for all experiments.

After we got these results we focused on our biLSTM-SA model and ignored all other models. We added a dropout layer to avoid overfitting and ran our loop to choose the best hyper parameters for this experiment. Table 7.6 shows the results of this new model we called drop-biLSTM-SA for simplicity, for all 3 experiments.

Experiment	*DEV*	*TEST*
2-way	86.14	85.92
3-way	85.58	85.15
4-way	81.26	79.83

Table 7.6: Achieved classification accuracy with drop-biLSTM-SA for all experiments

As we see in this table this new layer led to an improvement in results for all experiments especially for the 3-way-experiment.

After this improvement we decided to examine the effect of another tokenization technique hoping to gain even more accuracy. We tried to run this best model with sentence piece tokenization introduced in chapter 6 instead of the white space tokenization we used for all experiments until now. We applied this tokenization technique firstly for the 3-way-experiment in order to examine its effect before we generalize it to the rest of the experiments. Table 7.7 shows the classification accuracy, we achieved with this new model(we considered it as a separate model for simplicity) we called Bpe-drop-biLSTM-SA. As shown in this table, we achieved with this technique worse results than those with the previous model with white space tokenization, so, we did not examine this tokenization technique for the rest experiments (2-way and 4-way).

Model	*DEV*	*TEST*
Bpe-drop-biLSTM-SA	85.10	84.82

Table 7.7: Achieved classification accuracy with Bpe-drop-biLSTM-SA for 3-way-experiment

As we have seen the best results we achieved were with drop-biLSTM-SA model. We have mentioned at the beginning of this section that we have ran a loop for each

Parameter	$2-way$	$3-way$	$4-way$
Num of LSTM states	256	256	320
Num of LSTM layers	2	2	2
Embedding dimension	350	450	400
Drop-out value	0.4	0.4	0.4
Num of epochs	6	6	6

Table 7.8: Hyper parameter values with which we got the best results for the model drop-biLSTM-SA

experiment in order to get the perfect combination of hyper parameters. Table 7.8 shows the hyper parameter values with which we got the best results, for this model. Figures 7.2, 7.3, 7.4 show the confusion matrices of our best model "drop-biLSTM-SA" for the three experiments.

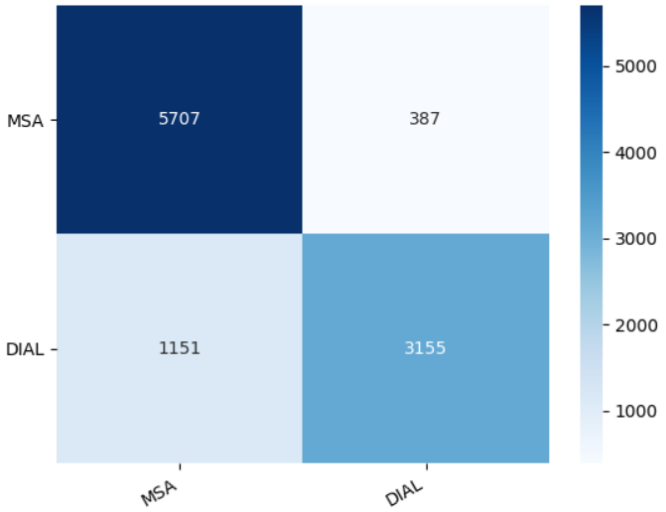

Figure 7.2: The confusion matrix for 2-way-experiment where the classes on the left represent the true dialects and those on the bottom represent the predicted dialects by drop-biLSTM-SA model

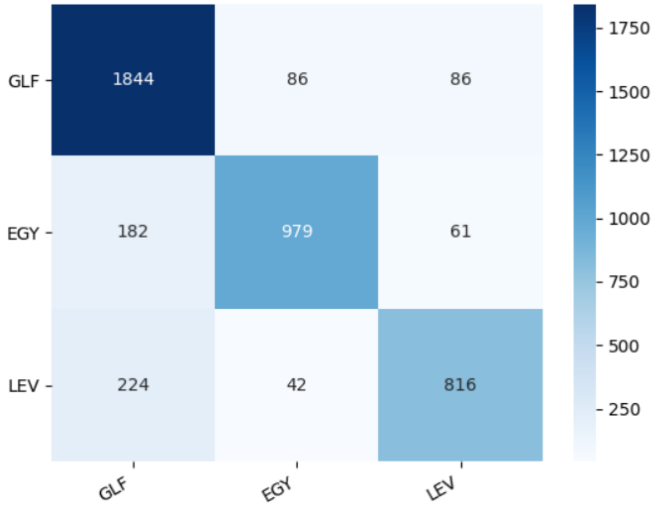

Figure 7.3: The confusion matrix for 3-way-experiment where the classes on the left represent the true dialects and those on the bottom represent the predicted dialects by drop-biLSTM-SA model

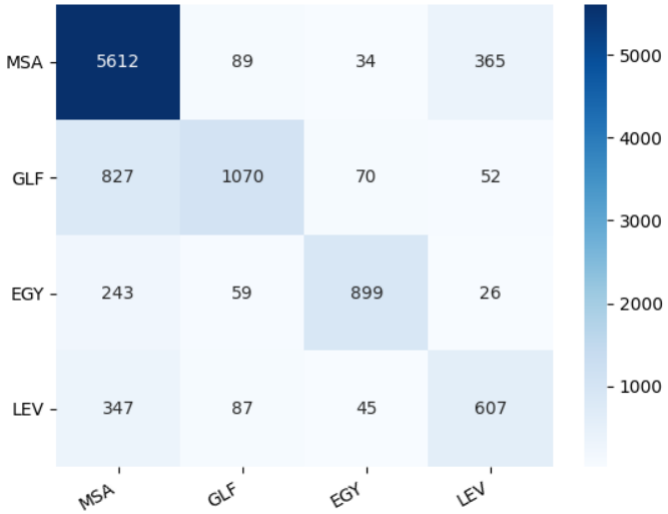

Figure 7.4: The confusion matrix for 4-way-experiment where the classes on the left represent the true dialects and those on the bottom represent the predicted dialects by drop-biLSTM-SA model

7.3 Error Analysis

In this section we analyse the behaviour of the best model "drop-biLSTM-SA" by trying to explore some patterns in its results. For each experiment we discuss the results achieved with this model in general then go more deeply and consider some individual errors hoping to find some interpretations enable us to give some recommendations could improve the accuracy in the experiments. Finally, we discuss the results obtained by all models, showing the effect of convolutional layers we added to biLSTM-SA model, as well as the effect of CBOW embedding we performed over the training data set, and at the end the effect of the sentence piece tokenization.

7.3.1 3-Way Experiment

As mentioned, for this experiment our models had to distinguish between three Arabic dialects, namely LEV, GLF and EGY. We discuss here the confusion matrix, in addition to some misclassified sentences produced by the best model for this experiment, namely the drop-biLSTM-SA model, as we have seen in the previous section. Considering the confusion matrix shown in figure 7.3, one can easily read that the easiest dialect to predict was the GLF dialect, that may be due to that fact that GLF dominants the data set for this experiment as shown in table 7.1. One can also read that it was really hard for the model to predict the LEV dialect correctly. For that, there is the following possible reasons: The big difference between the sub-dialects of the LEV dialect, for example between the Syrian and the Jordanian dialects. This lead to a big variance in the data of this dialect and of course to more difficulty by extracting Its distinctive features. In the following we give for each misclassified example its translation in English trying to clear the idea for non-Arabic speakers. The first example we consider is the following LEV sentence as test sample which were misclassified as EGY :

لو كان عنا لعيبة على مستوى متل دول اميركا اللاتينبة شو كان صار

In English "What happened, if we would have some professional players as in Latin American countries" Unfortunately, This sentence was misclassified as EGY, although it does not contain Egyptian words at all, and it can be categorized unequivocally as LEV. The main reason for this error is that all Arabic varieties even

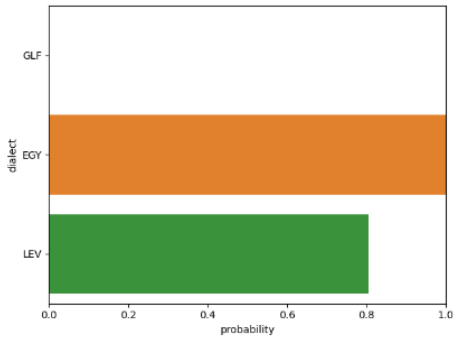

Figure 7.5: The probabilities for classifying the sentence in the first example

MSA are written now days without diacritical marks, although these marks can change the meaning entirely in some cases as in our considered sentence above. The word "دول", "*dwal*" in this sentence could have two completely different meanings.

The first one is with a diacritical mark "$fatha$" above the letter W where the word would be pronounced as "*dwal*" and means "countries" in all Arabic varieties . The second one is with a diacritical mark "*sokon*" above the letter W. In this case it would be pronounced as "*dwol*" which means "those" in EGY. Discovering the meaning in such cases is not difficult for Arabic speakers, and it is left for the reader to deduce it from the context usually. Unfortunately, it looks like that was very hard for our model to discover. But, what here was surprising to us is that the model predicted it as EGY with high confidence as shown in figure 7.5, which shows the probability, our model gave this sentence, for each dialect in this experiment. In this figure the probability for this sentence of being EGY represented with an orange color and that of being LEV represented with a green color .

Word	Pronunciation	Meaning
ده	*dah*	This "masculine"
مصر	*mesr*	Egypt
دي	*dih*	This "feminine"
دول	*dwol*	Those
الزمالك	*Alzamalek*	Football club name
زي	*zay*	Like
عايزين	*ayzeen*	We need

Table 7.9: The most repeated words for EGY in our training data set

After a careful review, we figured out that the word "*dwl*" in EGY belong to the 7 mostly repeated words of EGY in our training data set. Table 7.9 shows these words, which we called the most discriminative features of EGY in our data set. We are interested in these features, because they played a major role by classification in all our experiments and for all dialects as we will see later in this chapter. So, that interpret the behavior of the model for this sentence. It ignored some context information like the words order, the morphological structure and focused on the word "*dwl*". What made this problem even harder here is that the considered sentence does not have any word belong to the most discriminative features of LEV (Table 7.10).

Word	Pronunciation	Meaning
احنا	*ehna*	We
الوحدات	*Alwehdat*	Football club name
الفيصلي	*Alfaisaly*	Football club name
الاردن	*Alordon*	Jordan
مو	*mu*	Negation tool
اشي	*eshy*	Thing
بدون	*bdwn*	Without

Table 7.10: The most repeated words for LEV in our training data set

Finally, to check our interpretations we deleted the word "*dwal*" from the sentence and gave it again to the model to classify it. Unsurprising, it was predicted correctly as LEV. However, how far do these discriminative features affect the decisions of the model? To answer this question we consider the following sentence we gave to

our model to classify. This sentence contains two discriminative features for two different dialects:

سؤال يطرح نفسه بدون تحيز مادام الهلال بطل ليه طلع من الكأس بهالطريقة

In English"A question that remains unanswered, without prejudice: As long as the Al Hilal Club is a champion, why did it get out of the cup this way?"

This LEV sentence was classified as GLF although it contains one LEV discriminative feature which is the word بدون "*bdwn*" means in English "*without*" as shown in table 7.10. But it contains also the word الهلال "*alhelal*" which is a name of a Saudi football club and one of the discriminative features for GLF in our training data set (figure 7.11) with 505 occurrences (a large part of the comments in AOC data are about sport events). We think that our model got confused and misclassified this sentence as GLF due to the fact that the number of occurrences of the GLF discriminative feature "*alhelal*" is bigger than that for LEV discriminative feature "*bdwn*" (only 400). Figure 7.6 which shows the probabilities our model gave for those two dialects, where the probability of being GLF represented with blue and that of being LEV represented with green, enhance our thought. These converging possibilities for GLF and LEV shows how difficult it was for our model to predict the dialect for this sentence.

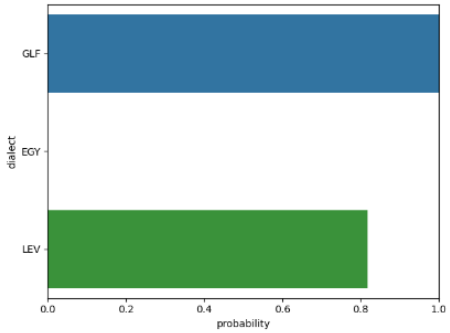

Figure 7.6: The probabilities for classifying the sentence in the second example

But the question is: Why our model failed to extract some other features in this sentence? To answer this question we try to extract these features manually. First, we consider the first part of the sentence سؤال يطرح نفسه "*soal yatrah nafsah*" in English "*the question arises*" which is an exemplary MSA morphological structure. Such MSA structures appear frequently in all dialectal varieties due to the huge overlap between them and MSA as we have seen in section 7.1. Since MSA is not involved as a class in this experiment, this part will play a neutral role in our opinion. The rest words in the sentence are either also MSA or common words in all dialects, so, we think they also have no big effect. But the last part بهالطريقة "*behaltariqa*" in English "*in this way*" has a very common LEV morphological structure. We think that, this LEV structure would be enough besides the discriminative feature "*bdwn*" to make the sentence correctly predictable. So, in our opinion, our model failed to discover it. There are two possible reasons hindered our model to extract such kind of features for this sentence: Either the small amount of training data available for the LEV or the high variation in LEV dialects as we mentioned earlier in this chapter, or the both.

Word	Pronunciation	Meaning
وش	*wsh*	What
االهلال	*Alhelal*	Saudi football club name
هذي	*hazy*	This
انتم	*Antom*	You
ابغى	*abgha*	I want
الحين	*Alheen*	Now

Table 7.11: The most repeated words for GLF in our training data set

7.3.2 2-Way Experiment

Considering the confusion matrix shown in figure 7.2 of our best model for this experiment, one can clearly read that, it was really hard for the model to distinguish the dialectal sentences. In other words, the number of dialectal sentences misclassified as MSA is significantly bigger than that for MSA sentences misclassified as dialect, although the test data set contains much more MSA sentences than DIAL sentences. In our opinion, the following possible reasons explain those results:

- MSA dominates the training data set for this experiment with about 15k sentences more than DIAL. This of course makes it easier for the model to extract more MSA features and recognize them later in the test phase.
- Dialectal sentences, unlike MSA have no standard which means less morphological structure, almost no grammatical rules. So, the kinds of features can be extracted from DIAL sentences is very limited as we have seen in the previous section where the most repeated vocabularies played the major role in the classification.
- As we have discussed earlier in this work, MSA considered as the base for all Arabic dialects. So, a huge part of DIAL sentences in our training data set contains MSA vocabularies or MSA structure...etc, and the reverse is not true. In addition to the fact that too many people used some MSA terms in their comments intentionally. This of course makes the possibility of classifying an DIAL sentence as MSA much greater than the possibility of classifying MSA sentence as a DIAL. Because the MSA features will be more clear comparing with DIAL features.

As an example for these three points we discussed, we consider the following dialectal sentence in our test data set which was classified as MSA:

الى اخي العزيز سيف ال خطاب صباحك قشطه اشتقت اليك اين انت

In English "To my dear brother Saif Al-Khattab, may you have a morning like a cream, I miss you, where are you" Actually this sentence consists entirely of MSA words, but the expression صباحك قشطه "*sabahuk keshta*" "*cream*" *morning*" used only in dialectal conversations. Although the two words construct it are pure MSA words when they would be considered separately. Capturing such kinds of features (such expressions) requires a vital knowledge in Arabic dialects, if they were not exist explicitly in the training data. Sometimes that is hard even for humans.

We have seen many reasons for misclassifying dialect sentences as MSA. In the following we discuss some MSA sentences were misclassified as DIAL:

بالنظر لاخر مواجهات اتوقع فوز الوحدات

In English "In view of the last (football) matches, I expect "$Al - Wehdat$" to win"

انه الفيصلي

In English "It's "$Al - Faisaly$""

خالد سعيد احترف على مقاعد الاحتياط في الزمالك

In English "Khaled Saeed got professional in the reserve seats in "$AL - Zamalek$""

Each of these three examples contains a dialectical mostly repeated words in out train data set. In the first two examples are the words الوحدات "$Al - Wehdat$" and الفيصلي "$Al - Faisaly$"". Both are names of Jordanian football clubs and have been repeated 767 and 531 times, respectively in our dialectal training data set (figure 7.12). In the third example also the word الزمالك "$AL - Zamalek$"" is a name of an Egyptian football club and have been repeated 631 times in our dialectal data set as figure 7.12 shows.

Word	Pronunciation	Meaning
ناس	Nas	People
االهلال	$Alhelal$	Saudi football club name
هذي	$hazy$	This
الوحدات	$Alwehdat$	Jordanian football club name
الزمالك	$Alzamalek$	Egyptian football club name
الفيصلي	$Alfaisaly$	Jordanian football club name
يمكن	$ymken$	May be

Table 7.12: The most repeated words for DIAL in our training data set

Considering the big effect of the most frequent words on the performance of our Model, as we have seen in the previous section, and due to the fact that these sentences are short and do not contain distinctive features for the MSA, it is not surprising that our model predicted them as DIAL.

7.3.3 4-Way Experiment

Given the confusion matrix for this experiment (figure 7.4), we can easily read that adding the new class MSA made the mission of distinguishing between dialects more difficult. Comparing this matrix with the confusion matrix of the 3-way experiment shown in figure 7.3 one can see that the number of correctly classified sentences for all 3 dialects GLF, EGY and LEV significantly decreased. This is not surprising given the discussion we had in the previous section. The most affected category is the GLF, as it lost about 800 sentences that were properly classified to be categorized as MSA. This is probably because the GLF is the closest dialect to MSA. What enhances this thought is the fact that most of the LEV sentences that were incorrectly classified as GLF sentences in the 3-way experiment were also incorrectly classified here but as MSA. LEV is the second most affected, losing about 200 correctly classified sentences in the 3-way experiment to be classified as MSA. The least affected is

EGY, which reinforces the prevailing theory that the furthest dialects from MSA is the Egyptian. In the following we discuss some sentences were classified correctly in the 3-way experiment and misclassified in the 4-way experiment as MSA.

مشهد غير حضاري ان كان من المواطنين الي وقفوا السيارات او المرور الي سحبها بالشكل

In English "An uncivilized scene, from both, the citizens who stopped the cars or traffic who pulled them like that"

This GLF sentence was misclassified as MSA. This sentence has only three features force any model to classify it as GLF, namely the following words الي "*elly*" "*which*" and it was repeated two times and the expression بالشكل "*blshakel*" "*like that*" and the word وقفوا "*wakafu*" "*stopped*". The rest of the sentence consists of MSA words, therefor, and with considering that those three mentioned features belong not only to GLF, but also to LEV and EGY which means they are no distinctive features for GLF. It is axiomatic that our model has classified it as MSA. Unfortunately, we face here almost the same problem we talked about in the beginning of this chapter, namely that people write now days without diacritical marks. Which leads that different words will be seen same in written form. Our case here is similar to that because these MSA words would be spoken differently in different dialects, but they have exactly the same shape in written form.

Finally, we consider the following GLF sentence which also classified correctly in 3-way experiment and as MSA in the 4-way experiment.

هنا تجد بعض اطباء تجميل الانف مثلا عاملين لهم شهره في الصحف على الفاضي ويمارسون شغلهم بالتعلم في الناس

In English "Here, you will find some cosmetic doctors, for example, who are unjustifiably famous in the newspapers and they are practicing their work with experiment on people"

This sentence does not contain GLF most repeated words, but it contains the word لهم "*lahom*" "*theirs*". This possessive adjective word is used only in GLF in standard form like in MSA. In 3-way experiment there is no MSA class, so it is axiomatic that our model has classified this sentence as GLF. But in 4-way-experiment it was classified as MSA, although it has another dialectal (but not mostly repeated) words like على الفاضي "*ala alfadi*" "*unjustifiably*". We think that due to the fact that MSA dominants the training data set for this experiment with about 50 k sentences versus 16 k for GLF, the sentences which have some shared words between GLF and MSA will be classified as MSA. Because, the number of repetitions of such words in MSA is bigger than that in GLF. This effect could be more worse, especially for GLF because MSA and GLF shared too many vocabularies like possessive adjectives, prepositions and demonstrative adjectives.

7.3.4 Overall Analysis and Discussion

In this section we interpret the results for the three experiments we performed by analysing the effects of the components we added to our baseline models in order to improve the achieved accuracy.

7.3.4.1 Effect of convolutional layers

As we have seen, adding these layers either did not lead to an improvement in accuracy of our model, as in the 3-way experiment, or made it worse, as in both 2-way and 4-way experiments. For this frustrating results, there are two possible reasons in our opinion. The first is the small amount of data that we have experimented with. The reason that led us to this explanation is that these convolutional layers according to the paper from which we were inspired by this idea led to improving the model's accuracy only when the amount of training data increased dramatically. The second possible reason is that the kind of data we have does not help these layers achieving their goal of extracting some association between neighboring words to reduce the dimension of features passed to the biLSTM layer. Such kind of associations in dialectal sentences is almost non-existent. Because dialectal sentences do not have a grammatical structure, for example, for these dialects there is no typical words order like in the MSA or in other languages.

7.3.4.2 Effect of Word2Vec CBOW embedding

Unfortunately, adding this embedding layer led to worse results for all experiments. Which means that instead of represents the words with numerical values in a way that reflects information about the language and about the similarities between those words, we got a representation even worse than that generated randomly. Actually, it is not surprising that such kind of information cannot be extracted over a very small data set, like our training data set. Extracting such information is not an easy task and requires a million of sentences instead of a few thousand we applied on. This lack of data given to this method may make it formed a confused numerical representation that does not reflect the true relationships between the words of the Arabic dialects. These incorrect representation made the task of feature extracting more difficult for our models. Another possible reason is the kind of these data as we have seen in the error analysis where a big part is about football games and sport, which lead surly to inaccurate language information.

7.3.4.3 Effect of SentncePiece Tokenizer

We have seen how this type of tokenization resulted in a slight reduction in accuracy for all experiments. This tokenization treats the entered sentence as a sequence of Unicode characters and applies the byte-pair-encoding (BPE) algorithm on it to encode repeated sub-words as separate units. In our opinion, this tokenization is not effective in the case of very closely related languages such as dialects. Because all Arabic varieties shared a huge amount of vocabularies and for a good portion of non-shared vocabularies, the difference is sometimes only in one or two letters. Because of that, these encoded sub-words could not be distinctive enough to classify the dialects. And the words itself would carry more information and represent more distinctive features.

Chapter 8

Conclusion

In this research work we performed a group of deep learning based experiments to identify the dialect of a given Arabic text. The main objective was to measure the performance of a group of deep learning models and attempting to improve the achieved classification accuracy using several deep learning techniques. We used for our experiments the AOC data set introduced in 7.1 and performed over it our three main experiments. First, the 2-way-experiment where the goal was to classify the given text in two classes, namely MSA and DIAL. The second one (3-way-experiment) was to distinguish between three dialects LEV, GLF and EGY. The last one (4-way-experiment) was to distinguish between the 4 mentioned Arabic varieties. At the end, we analyzed the errors made by the best model in details, in order to recognize some patterns in them.

In this work we have shown that the bi-directional LSTM with self-attention model introduced in 6.3 outperformed the two models RNN and the unidirectional LSTM (tables 7.3, 7.4, 7.5) especially when we added a drop-out layer to this model and increased the number of LSTM layers (table 7.6), which gave this model a high effectiveness and enable it to achieve the best accuracy.

We have also shown, that using convolutional layers with a purpose of reducing the dimensions of features before it will be passed on to LSTM layer, is not useful in the case of dialectal sentences, which do not have a grammatical structure and typical sentence order, and even led to worse results in some experiments.

With respect to CBOW embedding we can conclude that applying this technique on a small amount of data (as we did) led to extract inaccurate language information and giving an incorrect numerical representation to the words making the task more difficult for the model.

This work has also shown, that using the sentence piece tokenization technique with closely related languages such as the Arabic dialects, gave worse results than with white space tokenization (table 7.7). This is because encoding sub-words as separating unites lead to loss of the difference between many words that differ from each other by only a letter or two, which is very common in Arabic dialects.

Our error analysis has shown that one of the major challenges in identifying the dialect of a written Arabic text is the problem of writing without diacritical marks (section 7.3.1), as well as the fact that many words are pronounced differently from

one dialect to another, but are written in the same way. This leads to the loss of many of the features that can be extracted when the input text is in a speech form. This makes identifying the dialect of a written Arabic text more difficult than identifying this dialect in the case of this text is in a speech form.

The error analysis has also shown that the domination of the data set by one class confuses the model and leads to many sentences being incorrectly classified as this dominating class, since we have seen that many sentences that were correctly classified in the 3-way experiment , were misclassified as MSA in 4-way experiment (figure 7.4).

Another thing that can be deduced from our error analysis, is that the lack of the amount of training data and the concentration of a large part of it on a certain topic, led to the emergence of the problem of dominant words. Where some frequently repeated words played a major role in the classification and prevented the model from considering other types of features.

In this work, we could not reach the results obtained by the state of the art [11] for this problem on the AOC data set. In fact, most of these results were obtained using a pre-trained embedding applied on a very large data set (0.25 billion tweets). Therefore, our results that were obtained with a random embedding are not comparable to those results. In our work we were not able to use pre-trained embedding because we could not find valid embedding files for Arabic dialects in the web and we were limited in time, which prevented us from collecting a very large amount of data and applying any word embedding technique on it. Therefore, we will compare our results with those achieved in the state of the art, but with random embedding which are an accuracy of 85.23% for the 2-way-experiment and 85.93% for the 3-way-experiment and 80.21% for the 4-way-experiment. From our best results shown in table 7.6 one can see that we got better accuracy for the 2-way-experiment and a little less accuracy for the rest two experiment. Our recommendations to improve the achieved accuracy in light of these results and error analysis are: First, to use a larger volume of training data as well as with more diverse topics, to avoid the problem of dominant words, and to enable the model to extract new types of features that can solve a problem of diacritical marks, such as morphological structure or the typical sentence order. Second, is to use a pre-trained embedding trained on a very large amount of data to give a thoughtful correct numerical representation to the vocabularies reflects the similarity between them.

Bibliography

[1] Mohamed Ali. "Character level convolutional neural network for Arabic dialect identification." In: *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. 2018, pp. 122–127.

[2] Areej Odah O Alshutayri. "Arabic Dialect Texts Classification." PhD thesis. University of Leeds, 2018.

[3] Maha J Althobaiti. "Automatic Arabic Dialect Identification Systems for Written Texts: A Survey." In: *arXiv preprint arXiv:2009.12622* (2020).

[4] Ibrahim Bassal. "Hebrew and Aramaic Elements in the Israeli Vernacular Christian Arabic and in the Written Christian Arabic of Palestine, Syria and Lebanon." In: *The Levantine Review* 4.1 (2015), pp. 86–116.

[5] Yoshua Bengio, Patrice Simard, and Paolo Frasconi. "Learning long-term dependencies with gradient descent is difficult." In: *IEEE transactions on neural networks* 5.2 (1994), pp. 157–166.

[6] Houda Bouamor et al. "The madar arabic dialect corpus and lexicon." In: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. 2018.

[7] William B Cavnar, John M Trenkle, et al. "N-gram-based text categorization." In: *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*. Vol. 161175. Citeseer. 1994.

[8] Corinna Cortes and Vladimir Vapnik. "Support-vector networks." In: *Machine learning* 20.3 (1995), pp. 273–297.

[9] Kareem Darwish. "Arabizi detection and conversion to Arabic." In: *arXiv preprint arXiv:1306.6755* (2013).

[10] Mona Diab et al. "COLABA: Arabic dialect annotation and processing." In: *Lrec workshop on semitic language processing*. 2010, pp. 66–74.

[11] Mohamed Elaraby and Muhammad Abdul-Mageed. "Deep models for arabic dialect identification on benchmarked data." In: *Proceedings of the Fifth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2018)*. 2018, pp. 263–274.

[12] Mohamed Eldesouki et al. "Qcri@ dsl 2016: Spoken arabic dialect identification using textual features." In: *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*. 2016, pp. 221–226.

[13] Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. "AIDA: Identifying code switching in informal Arabic text." In: *Proceedings of The First Workshop on Computational Approaches to Code Switching*. 2014, pp. 94–101.

[14] Heba Elfardy, Mohamed Al-Badrashiny, and Mona Diab. "Code switch point detection in Arabic." In: *International Conference on Application of Natural Language to Information Systems*. Springer. 2013, pp. 412–416.

[15] Heba Elfardy and Mona Diab. "Sentence level dialect identification in Arabic." In: *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. 2013, pp. 456–461.

[16] Jeffrey L Elman. "Finding structure in time." In: *Cognitive science* 14.2 (1990), pp. 179–211.

[17] Rong-En Fan et al. "LIBLINEAR: A library for large linear classification." In: *Journal of machine learning research* 9.Aug (2008), pp. 1871–1874.

[18] Björn Gambäck and Amitava Das. "On measuring the complexity of code-mixing." In: *Proceedings of the 11th International Conference on Natural Language Processing, Goa, India*. 2014, pp. 1–7.

[19] Alex Graves. "Generating sequences with recurrent neural networks." In: *arXiv preprint arXiv:1308.0850* (2013).

[20] Chinnappa Guggilla. "Discrimination between similar languages, varieties and dialects using cnn-and lstm-based deep neural networks." In: *Proceedings of the Third Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial3)*. 2016, pp. 185–194.

[21] Nizar Habash, Houda Bouamor, and Kemal Oflazer. "A Multidialectal Parallel Corpus of Arabic." In: (2014).

[22] Nizar Habash, Mona T Diab, and Owen Rambow. "Conventional Orthography for Dialectal Arabic." In: *LREC*. 2012, pp. 711–718.

[23] Nizar Habash et al. "Guidelines for annotation of Arabic dialectness." In: *Proceedings of the LREC Workshop on HLT & NLP within the Arabic world*. 2008, pp. 49–53.

[24] Sepp Hochreiter and Jürgen Schmidhuber. "Long short-term memory." In: *Neural computation* 9.8 (1997), pp. 1735–1780.

[25] Beakcheol Jang et al. "Bi-LSTM model to increase accuracy in text classification: combining Word2vec CNN and attention mechanism." In: *Applied Sciences* 10.17 (2020), p. 5841.

[26] Thorsten Joachims. *Making large-scale SVM learning practical*. Tech. rep. Technical Report, 1998.

[27] Nikhil Ketkar. "Convolutional neural networks." In: *Deep Learning with Python*. Springer, 2017, pp. 63–78.

[28] Jihun Kim and Minho Lee. "Robust lane detection based on convolutional neural network and random sample consensus." In: *International conference on neural information processing*. Springer. 2014, pp. 454–461.

[29] David G Kleinbaum et al. *Logistic regression*. Springer, 2002.

[30] Taku Kudo and John Richardson. "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing." In: *arXiv preprint arXiv:1808.06226* (2018).

[31] Zhouhan Lin et al. "A structured self-attentive sentence embedding." In: *arXiv preprint arXiv:1703.03130* (2017).

[32] Dennis V Lindley. "Fiducial distributions and Bayes' theorem." In: *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), pp. 102–107.

[33] Shervin Malmasi, Eshrag Refaee, and Mark Dras. "Arabic dialect identification using a parallel multidialectal corpus." In: *Conference of the Pacific Association for Computational Linguistics*. Springer. 2015, pp. 35–53.

[34] Paul McNamee. "Language identification: a solved problem suitable for undergraduate instruction." In: *Journal of computing sciences in colleges* 20.3 (2005), pp. 94–101.

[35] Michael Frederick McTear, Zoraida Callejas, and David Griol. *The conversational interface*. Vol. 6. 94. Springer, 2016.

[36] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality." In: *arXiv preprint arXiv:1310.4546* (2013).

[37] Tomas Mikolov et al. "Efficient estimation of word representations in vector space." In: *arXiv preprint arXiv:1301.3781* (2013).

[38] TM Mitchell. "Machine Learning, McGraw-Hill Higher Education." In: *New York* (1997).

[39] Arfath Pasha et al. "Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic." In: *Lrec*. Vol. 14. 2014. 2014, pp. 1094–1101.

[40] Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur. "A time delay neural network architecture for efficient modeling of long temporal contexts." In: *Sixteenth Annual Conference of the International Speech Communication Association*. 2015.

[41] Jeffrey Pennington, Richard Socher, and Christopher D Manning. "Glove: Global vectors for word representation." In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[42] Stephen Robertson. "Understanding inverse document frequency: on theoretical arguments for IDF." In: *Journal of documentation* (2004).

[43] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors." In: *nature* 323.6088 (1986), pp. 533–536.

[44] Fatiha Sadat, Farzindar Kazemi, and Atefeh Farzindar. "Automatic identification of arabic language varieties and dialects in social media." In: *Proceedings of the Second Workshop on Natural Language Processing for Social Media (SocialNLP)*. 2014, pp. 22–27.

[45] Mohammad Salameh, Houda Bouamor, and Nizar Habash. "Fine-grained arabic dialect identification." In: *Proceedings of the 27th International Conference on Computational Linguistics*. 2018, pp. 1332–1344.

[46] Younes Samih and Laura Kallmeyer. "Dialectal Arabic processing Using Deep Learning." PhD thesis. Heinrich-Heine-Universität Düsseldorf, 2017.

[47] Younes Samih and Wolfgang Maier. "Detecting code-switching in moroccan Arabic social media." In: *SocialNLP@ IJCAI-2016, New York* (2016).

[48] Younes Samih et al. "Multilingual code-switching identification via lstm recurrent neural networks." In: *Proceedings of the Second Workshop on Computational Approaches to Code Switching*. 2016, pp. 50–59.

[49] Mike Schuster and Kuldip K Paliwal. "Bidirectional recurrent neural networks." In: *IEEE transactions on Signal Processing* 45.11 (1997), pp. 2673–2681.

[50] Rico Sennrich, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." In: *arXiv preprint arXiv:1508.07909* (2015).

[51] Tony C Smith and Eibe Frank. "Introducing machine learning concepts with WEKA." In: *Statistical genomics*. Springer, 2016, pp. 353–378.

[52] Andreas Stolcke. "SRILM-an extensible language modeling toolkit." In: *Seventh international conference on spoken language processing*. 2002.

[53] Shahadat Uddin et al. "Comparing different supervised machine learning algorithms for disease prediction." In: *BMC Medical Informatics and Decision Making* 19.1 (2019), pp. 1–16.

[54] Alex Waibel et al. "Phoneme recognition using time-delay neural networks." In: *IEEE transactions on acoustics, speech, and signal processing* 37.3 (1989), pp. 328–339.

[55] Wikipedia. *Convolutional neural network*. 2021. URL: https://en.wikipedia.org/wiki/Convolutional_neural_network.

[56] Wikipedia. *Deep learning*. 2018. URL: https://simple.wikipedia.org/wiki/Deep_learning.

[57] Omar Zaidan and Chris Callison-Burch. "The arabic online commentary dataset: an annotated dataset of informal arabic with high dialectal content." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. 2011, pp. 37–41.

[58] Omar F Zaidan and Chris Callison-Burch. "Arabic dialect identification." In: *Computational Linguistics* 40.1 (2014), pp. 171–202.