Media Engineering and Technology Faculty
German University in Cairo

Arabic Part of Speech Tagger

Bachelor Thesis

Author:	Aisha Mohammed El-Badrawy
Supervisors:	Dr. Mohamed Elmahdy
Submission Date:	15 May, 2017

Media Engineering and Technology Faculty
German University in Cairo

Arabic Part of Speech Tagger

Bachelor Thesis

Author:	Aisha Mohammed El-Badrawy
Supervisors:	Dr. Mohamed Elmahdy
Submission Date:	15 May, 2017

This is to certify that:

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

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

Aisha Mohammed El-Badrawy
15 May, 2017

Acknowledgments

"All the praises and thanks be to Allah, Who has guided us to this, never could we have found guidance, were it not that Allah had guided us"

Quran [7:43]

I would like to express my gratitude toward my supervisor, Dr. Mohamed Elmahdy as he showed a great deal of tolerance, guidance, motivation and mentor-ship throughout this project, this paper would not have been what it is, had he been not helping me through each step of the way, steering me in the right direction.

I would also like to thank my Family and Friends for supporting me throughout this challenging and rewarding journey.

Abstract

The demanding increase in applications that understand natural human language lead to the popularity of Natural Language Processing (NLP). Many of the applications that we use on a daily basis incorporate NLP, from simple tasks such as automatic text correction to speech recognition. With research being done on NLP of English language but not much attention given to the NLP of Arabic language, despite being it a rich language spoken by around 400 million speakers. Thus, the purpose of this paper is to implement a supervised Arabic Part of Speech (POS) Tagger, as POS tagging is an important task in NLP, that serves as a building block for more advanced tasks, as well as experimenting with different machine learning algorithms, namely Naïve Bayes, Support Vector Machines (SVM), Random Forest and Neural networks. Comparing their performances in the process, to find the algorithm with the highest accuracy, and present an Arabic POS Tagger with state of the art results.

Contents

Chapter 1

Introduction

1.1 Motivation

Arabic is a language spoken by around 400 million people, making it one of the most spoken languages in the world. Arabic is the official language in 22 countries dominantly lying in the middle east and north Africa, it is a rich language with cultural, religious and political importance. Still research interest in processing Arabic language compared to English language is very limited.

With the rapid increase in globalization, languages are becoming a key part in technology, with wide variety of applications and tools for translation, speech recognition and information retrieval, as well as the increase in Arabic language presence in the technology and social media scene, thus research in Arabic language processing should be of an importance to keep up with recent technology.

1.2 Aim of Work

The aim of this thesis is to implement a Supervised Arabic Tagger that produces state of the Art results using different traditional machine learning techniques as Naïve Bayes, Support Vector Machines (SVM) as well as exploring neural networks and experimenting with them, since there are not many Arabic Neural taggers.

This Tagger is implemented using Python as a programming language and several machine learning libraries that model machine learning algorithms.

1.3 Thesis Outline

This Thesis consists of five Chapters, Chapter 2: "Background", includes theoretical details regarding Natural Language Processing, Part of Speech tagging and machine learning. Chapter 3:"Methodology", is the chapter that describes the actual system and how it is implemented, while Chapter 4:"Experiments and Evaluation", is where different experiments using various machine learning algorithms are carried out and a comparison of their performances is made. Chapter 5 is the conclusion and chapter 6:"Limitations and Future Work", expresses the limitations faced implementing this Tagger and discusses future work.

Chapter 2

Background

2.1 Natural Language Processing

Natural Language Processing (NLP) is a widely growing field that deals with human language-computer interactions, in other words any language manipulation done by the computer, NLP is mainly concerned with understanding and generating text. NLP has many tasks, which are very closely related, some task serve as real world applications while others serve as subtask that contribute to larger applications, the tasks fall under four main criteria, syntax, semantic, discourse and speech, below is an example from each category:

- Parsing (Syntax): Parsing is the task of syntactically analyzing text, a parser is a program that figures out the grammar structure of a sentence, and decides if a word is a subject or an object depending on a specified grammar.
- Word Sense Disambiguation (Semantic): This task when given a sentence understands the meaning of the word depending on its context, taking into consideration the semantics of the sentence, thus falling under the semantic category.
- Automatic Summarization (Discourse): Discourse is to speak or write about a topic, in our case automatic summarization is writing about a certain topic, given a document, it generates a few lines to summarize it.
- Speech Recognition (Speech): It falls under speech category as we can see it is related to speech, it allows the computer to understand human's speech, which is one of the most challenging task in NLP.

NLP is considered as a hard problem in computer science, even though languages are very easy to understand to humans, when it comes to computers they have to be provided with large set of rules or rely on machine learning algorithms to derive patterns and define relations between words, since languages are very ambiguous.

2.2 Part of Speech Tagging

POS Tagging is an NLP syntactic task, that assigns a word it's corresponding part of speech tag, which is also known as word class, these tags that are used to identify words are called a tag set, a tag set can be as general as describing (Noun,Verb,Adverb) and as detailed as describing (male or female, singular or plural, past or present), language is ambiguous and therefore a word may have multiple tags, deciding between them makes POS a hard task, for example the following sentences "I will play" and "the play was over", the word play in the first sentence is a Noun while in the second one it is a Verb.

Part of Speech Tagging is the problem of assigning tags, using a defined Tagset to un-tagged new words given a previously tagged Corpus.

2.2.1 Corpus

A Corpus was defined by leech in 1992 as "a large collection of natural language material stored in machine readable form that can be easily accessed, automatically searched, manipulated, copied and transferred" [17].

A Corpus provides the POS tagger with the necessary knowledge, to deal with language ambiguity without prior linguistic knowledge, the first English printed corpus to be developed was the Brown corpus, in 1967 with 1 million words developed at Brown university, it played an impotent role in the Advent of computational linguistics and NLP tasks. There is no free corpus for the Arabic language, a few number of corpus is available for Arabic, compared to the ones available for English, a well know Arabic corpus, is the Penn Arabic Treebank (PAT), which was developed at University of Pennsylvania in 2001 [2].

2.2.2 Tagset

A Tagset is all the possible tags for words in a language, where each word has it's corresponding tag, there are various tagsets for English with the most popular being Penn Treebank tagset with 31 tags. There are several Arabic tagset developed for tagger systems, like Shreen khoja's APT system with 177 detailed tags, El-Kareh and Al-Ansary tagset which consists of 77 tags and Al-Shamsi and Guessom tagset which contains of 55 tags used for their Hidden Markov Model (HMM) tagger [2].

The size of the tagset affects the performance of tagger, with a detailed tagset the performance is bound to decrease, but increase with smaller general tagsets, therefore the structure of the tagset must be taken into consideration when comparing tagging systems [1].

2.2.3 Importance of POS Tags

Knowing the POS tag of a word gives a lot of information about the word itself as well as neighbouring words, for instance words with POS tag <noun> are morel likely to be preceded by determiners or adjective, while words with POS tag <verb> are preceded by nouns, thus, making POS as a subtask in many NLP tasks like, information extraction,text summarization and machine translation [11].

2.2.4 Tagging Techniques

Tagging techniques are divided mainly to four categories, Rule-Based stochastic, Hybrid and Neural Taggers [1] as listed below:

- Rule-Based Taggers: Rule Based tagging revolves around a set of rules which is used to determine the tag of a word using its context based on grammatical or morphological rules.
- Stochastic Taggers: Stochastic tagging is based on algorithms that uses frequency an probability from training corpus.
- Hybrid Taggers: Hybrid Taggers are Taggers that uses different types of taggers simultaneously, by combining Rule-Based tagging with Stochastic tagging.
- Neural Taggers: With the increasing popularity of neural networks, neural tagging emerged and it uses neural networks for tagging.

2.3 The Arabic Language

The Arabic language has multiple variants, one of which is Modern Standard Arabic (MSA), which is the formal Arabic language used for media and education across the Arab world, other variants are the Arabic dialects which are used for daily communication only, this linguistic situation is nearly the same with other languages except that it differs in exactly two points :

- The great difference between standard Arabic and dialects.
- The fact that standard Arabic is not a native language for any Arab.

MSA is the official language for the Arab world, it is syntactically and morphologically based on classical Arabic, which is the language of the Quran, while dialects are the native Arabic language used for every day communications,it is neither standardized nor used in education, even though there is a rich culture of movies, literature and folktales, but that is changing as Arabs are becoming more involved in electronic media [8].

2.4 Related Work

Researches have realized that POS tagging is a basic and intermediate task in many other NLP problems, and that is why a lot of research was done to improve POS Tagging across the last decades. The Arabic Language is a rich morphological language that syntactically differs from other Indo-European Languages like English, so even though many research have been done regarding English POS tagging, not many have been done for Arabic [2].

In this Section, a brief history of both English and Arabic POS tagging is going to be discussed.

2.4.1 English POS Tagging

POS English Taggers			
Year	Author/s	Approach	Accuracy %
1963	Klein and Simmons	Rule-Based	90
1965	Stolz	Probabilities	92
1971	Greene and Rubin	Rule-Based	77
1976	Bahl and Mercer	Probabilities	98.6
1987	Garside	Probabilities and Rule-Based	97
1987	Church	Probabilities	95-97
1988	Derose	Probabilities	96
1989	Benello	Neural Networks	95
1992	Cutting	Probabilities	96
1992	Brill	Rule based	96
1992	Kupiece	Probabilities	96
1993	Weischedele	Probabilities	94
1994	Schmid	Neural Networks	96
1994	Tapanainen and Voultilainen	Probabilities and Rule based	98
2004	Prins	Probabilities	93.62
2007	Tamburini	Probabilities and Transformation-Based	96
2009	Zanoli and Pianta	Rule-Based	96.06
2011	Sogard	New Algorithm	97.50

Table 2.1: This Table shows history of English POS Taggers stating year, Author, the Approach as well as the Accuracy.

NLP systems were implemented in the 60's, but POS tagging was part of these systems and not a separate entity. Tagging techniques have changed across the decades, from rule based taggers in the 60's, to taggers applying statistical and probabilistic approaches in the 70's and 80's, with new techniques introduced in 90's, by applying machine learning algorithms and neural networks as well as combining different approaches, in hope to achieve better results. In Table 2.1 a list of POS English taggers are presented together with the the author, accuracy and approach followed [2].

2.4.2 Arabic POS Tagging

POS Arabic Taggers			
Year	Author/s	Approach	Accuracy %
2000	El-Kareh and Al-Ansary	Hybrid	90
2001	Shreen Khoja	Hybrid	86
2002	Maamouri and Cieri	Rule-Based	96
2004	Mona Diab	Support Vector Machine and Rule based	95.49
2004	Banko and Moore	Statistical	97
2006	Tlili-Guiassa	Hybrid	86
1989	Shamsi and Guessoum	Statistical	97
2008	Shihadeh Alqrainy	Rule-Based	91
2014	Elhadj	Statistical	96

Table 2.2: This Table shows history of Arabic POS Taggers stating year, Author, the Approach as well as the Accuracy.

Several Arabic tagging systems were developed, in Table 2.2, Arabic taggers, their accuracy and the techniques behind them are listed.

El-Kareh and Al-Ansary tagger was based on derived traditional Arabic grammar, it used both statistical methods and morphological rules in the from of HMM, and that is why it is a hybrid tagger, it yields a 90% accuracy. Shreen khoja developed a tagging system, the Automatic Arabic POS-Tagger (APT), it is considered as the first Arabic tagging system, it uses both statistical and rule-based approaches, with 131 tags derived from Arabic grammar, it yields an accuracy of 86%. Maamouri and Cieri's Tagger was a rule-based tagger based on the annotated output produced by morphological analyzer Tim Buckwalter. Mona Diab developed an Arabic tagger using Support Vector Machines which consisted of 24 tags giving an accuracy of 95.49%. Banko and Moore present a Arabic statistical HMM tagger giving an accuracy of 97%. Guissa used machine learning memory based methods together with statistical approach producing a hybrid tagger with an accuracy of 86%. Shamsi and Guessom were one of the few researchers to properly consider the structure of the Arabic sentence, they developed an Arabic tagger for un-vocalized text based on HMM. Al-Garainy developed a rule based tagger called Arabic

Morpho-syntactic Tagger (AMT), this tagger assigned POS tags to un-tagged corpus using pattern, contextual and lexical rules with an accuracy of 91%. Elhadij developed an Arabic tagger based on HMM also considering the structure of Arabic sentence yielding an accuracy of 96% [2].

2.5 Machine Learning

Learning is a necessary human function that requires change in order to become better at a given task under certain circumstances, learning is usually defined as the process to gain knowledge and skills, with the exception of Rote learning, which is just learning by memorizing without detecting patterns in the given information, thus we could define Machine Learning as the computer science filed where it is concerned with computational modeling of learning [15].

2.6 Learning Paradigms and Techniques

The learning process consists of a learner and a teacher, the teacher knows information required to perform a certain task, while the student needs to acquire this information to perform this task. There are different learning techniques, one of them is learning by example which is the core learning paradigm for Machine learning, there are three types of Machine Learning techniques, Supervised, Unsupervised and Reinforcement Learning [15].

2.6.1 Supervised Learning

The machine Learning algorithm is given a labeled data set, it just learns the mapping between the data and the labels, then it can solve some type of task.

2.6.2 Unsupervised Learning

The machine Learning algorithm is given an unlabeled data set, so it has to learn by itself what the structure of the data is to solve some given task, this is hard to do, but more convenient since getting hold of a labeled set is often hard and most of the data is unlabeled.

2.6.3 Reinforcement Learning

Reinforcement Learning is inspired by the idea that we learn from the interaction with our environment, if for example a baby is left in an environment with no one to guide and no idea of what it should do, this will create a learning process through cause and effect, and consequence of actions, Reinforcement Learning is a goal directed learning paradigm, where an agent is subjected to an environment and is not told what do do, but thorough a series of actions it figures out what to do, if an action yields a reward that means it is going in the right direction as shown in Figure 2.1, it is a process of trail and error [34].

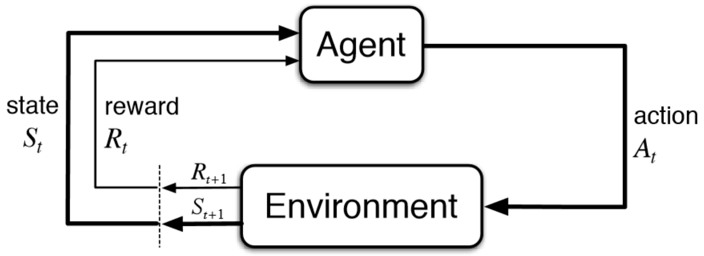

Figure 2.1: Reinforcement Learning Cycle

2.7 Artificial Neural Networks

Artificial Neural Network (ANN) are machine learning models inspired by the biology of the human brain, since the brain is responsible for controlling the human body through reacting and transmitting signals, scientist have always been fascinated by inner workings of the brain, how it recognize faces and finds patterns and how it controls the human body, to be able to model something like it, that would be able performs such task would be very powerful and a doorway to endless possibilities.

The way neurons are connected together in a neural networks defines it's structure and affects its output, neural networks are either recurrent or non-recurrent, non-recurrent networks are feed forward neural networks, meaning that the signal passes in one direction from input layer to output layer, while in recurrent networks the signal can go in both directions introducing cycles [30].

2.7.1 Evolution of Neural Networks

The first neuron model was presented in 1943 by Mculloch and Pitts [19], where a neuron could take a number of inputs, sum them and if their sum exceeds a certain threshold it outputs a ”1”, else it outputs a ”0”, it could model basic logical operations, but it lacked the desired attribute, the ability to learn .

In 1958 Frank Rosenbelt built on Mculloch and Pitts model and developed the perception [29], to make the model learn, he added weights to the inputs, so the sum became the sum of product of inputs and their weights, the model was presented with train data samples, weights were randomly assigned at first, it's output was compared to the expected output, if the expected output was 0, but Perceptron output was 1, decrease the weight of input that was 1, if the expected output was 1 but the Perceptron yielded 0, increase the weight of the input that was 1, this process of comparing outputs is repeated for all training sample until Perceptron produces the correct output.

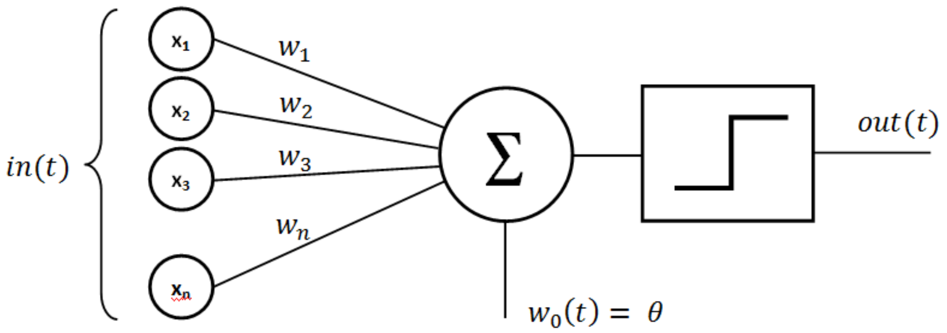

Figure 2.2: The Perceptron, Fundamental unit of Neural Network

2.7.2 Multi-layer Perceptron and Back-propagation

One of the Perceptron limitations was the fact that it had one output, so increasing the number of outputs would solve this problem, this was done by introducing multiple Perceptrons, all the input nodes of Percerptrons acting as a single input layer, and all output nodes serving as one output layer, where the input layer receives the signal and each node is responsible for part of the output. The intuition behind ANN was to build a network of interconnected neurons (Perceptrons) with many layers, not just one output layer, so instead of passing the input directly to output layer, it passes through a hidden layer, which could be one or more layers, where the output of each layer serves as an input to following one, till it reaches the output layer as shown in Figure 2.3.

Making such a model learn was impossible, since there was no way we could compare the output of a hidden layer to an expected output as we can not know what is the right output of hidden layer beforehand. The solution to this problem was the chain rule, first instead of limiting the Perceptron's output to either a ”1” or ”0” through a threshold, a nonlinear differentiable activation function, like sigmoid is used that ranges the output from 0.0 till 0.9, thus this function was used to adjust the weights of network to minimize error, but still the problem of updating weights of hidden layer was not solved, until back-propagation was introduced, back-propagation is the task of updating the weights of layers in a network by propagating backwards in the network, using the chain rule on, the differentiable activation function.

2.7.3 Recurrent Neural Networks

Unlike feedforward networks that are concerned about the current input, and the output of a certain input is not affected by the output of a previous one, Recurrent Neural

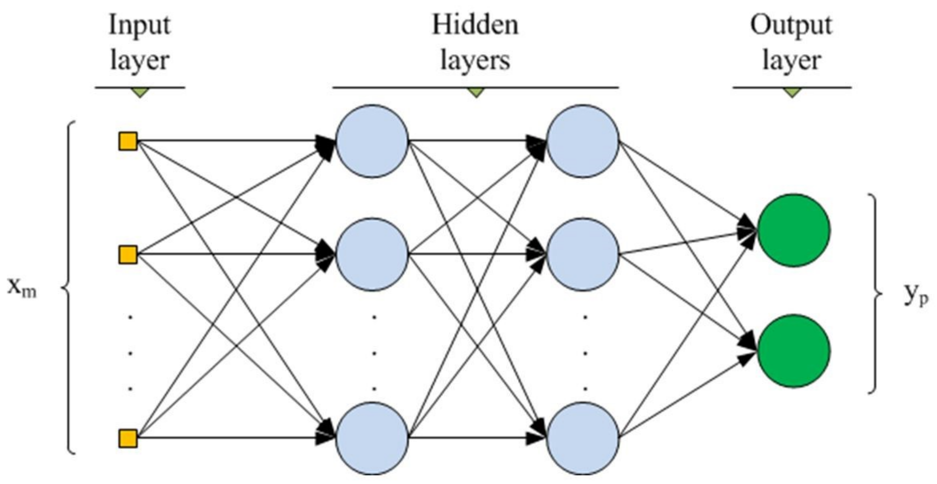

Figure 2.3: Multi-Layer Perceptron Neural Network

Networks (RNN) remember the outputs of previous inputs, and so it affects the decision of new input, due to the fact that it has loops, which enables them to have a form of memory, which made them a very popular language and speech models. In Figure 2.4 we can see the node with weight going back to it, and a small box expressing time steps t, once it is unfolded it is seen as a feed forward network with inputs having the same weight, where each of sequence input as a separate input in each stage, back-propagation algorithm is applied to the unfolded version. Recurrent networks fell short when it came to remembering sequence for a long time, and that is when the idea of augmenting RNN with explicit memory surfaced, a special unit called Long Short Term Memory (LSTM), which acted like a gated neuron, was able to remember for a long time, soon after, the LSTM proved to be more effective than conventional RNNs [16].

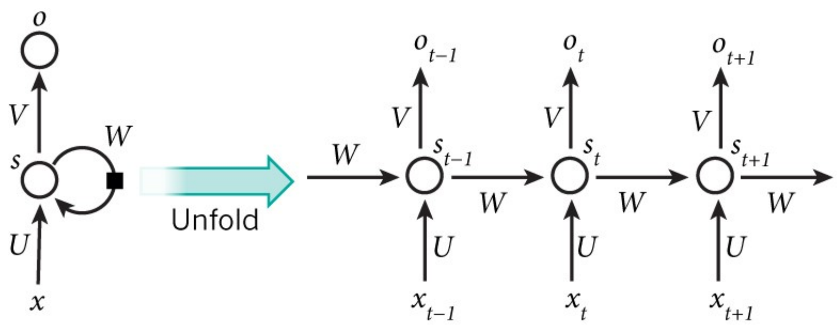

Figure 2.4: A RNN LSTM node unfolding with time t

2.7.4 Neural Networks Paradigms

There are multiple Neural Networks paradigms as displayed in Figure 2.5 above, the first layer from the bottom is the input layer and the upper layer is the output layer, the layer

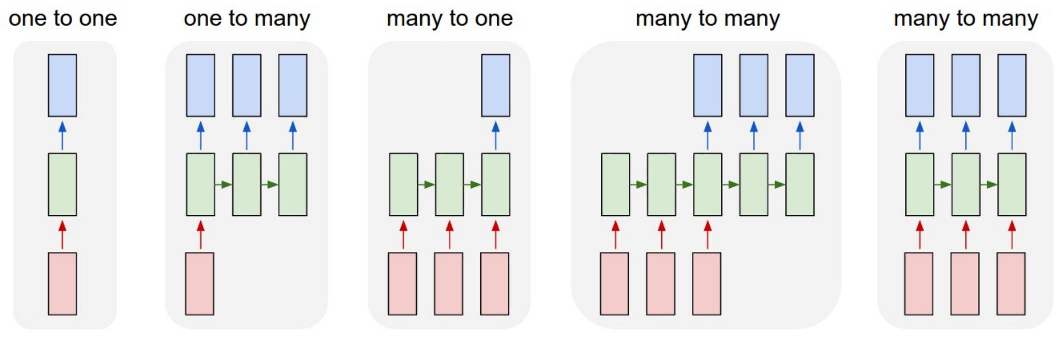

Figure 2.5: Neural Networks Paradigms

in the middle is the hidden layer, it can be observed that both input and output layer have fixed size, while units of hidden layer are variable.

First neural network from the left, the one to one, is the basic neural network architecture, where given an input vector it returns a fixed size output, for example given a word, it outputs the corresponding tag.

The one to many architecture, is where a fixed input is given to a network and it outputs a variable length output, for example the task of image captioning, where given a single image, a variable sequence of words is outputted.

The many to one paradigm, is where a variable length input is given and a single label is outputted, a machine learning task that applies to this paradigm would be sentiment analysis, since it classifies a sequence of words of variable length as either positive or negative.

The many to many paradigm from the right is a synced sequence to sequence architecture used for task such as video classification, where we want to classify each frame in a video as either an add or news or sport and so on.

The second many to many paradigm from the right, is the sequence to sequence (seq2seq), which is used greatly with NLP tasks, such as translation, text generation and POS tagging, as it takes as an input a sequence of words and outputs a sequence of labels, each label corresponding to the a word in the sequence in case of POS tagging.

2.7.5 Sequence to Sequence Architecture

Even though traditional deep learning networks achieved great results, it was restricting as it required both the input and the output to be of fixed dimensionality, and thus failing to work with machine learning tasks that are best mapped to sequence problems, like speech recognition and machine translation. Using stacked layers of LSTM RNN networks, it is possible to achieve sequence to sequence modelling [33].

2.8 Deep Learning

It is a sub-field in machine learning, that revolves around artificial neural networks and utilizes them to achieve outstanding results. It can also be viewed as just a neural network with multiple layers.

Conventional machine learning techniques fell short when it came to processing raw data, prepossessing the data had to take place, using feature extraction which needed engineering expertise and wide knowledge of the data set being used, then these features would have to be transformed to a certain representation or feature vectors, on the other hand, in deep learning no such task is needed, as a deep learning network, takes as an input raw data and through passing data through multiple layers, resulting into multiple transformations with each new transformation becoming more accurate, by doing so learning by itself the patterns, which is known as representation learning. What makes deep learning so fascinating, is the fact that these features are not engineered by humans but self learnt by a general learning procedure [16].

2.9 Classification

Classification can be found in many human activities, generally it covers any context that involves making decisions based on available information, a classification procedure is doing the same when faced when new information, formally it is the task of constructing a classification procedure (classifier) from a set of data which its true correct classes are already determined [20].

Classification in Machine learning is supervised, when given labeled data, but when unsupervised it is called clustering, Machine learning aims to generate algorithms that resemble human reasoning by learning from a series of examples [20].

Supervised Machine learning is applied to real world problems, by first training a classifier with labeled data, so it can later generalize the problem when presented with unseen data.[13].

2.10 Classifiers

A Classifier is a function that assigns a class label to an example, where an example is a tuple of features E(x1, x2, x3, ..., xi) [36].

2.10.1 Naïve Bayes

Is a statistical machine learning technique, based on a probabilistic model, in Naïve Bayes classifier each feature plays a role in determining the right label for a given input, as Naïve

Bayes classifier calculates prior probability of each feature using the frequency of each feature in the input data provided, then it multiples this probability with contribution of each feature to produce the likelihood estimate, the feature with the highest likelihood estimate determines the right label as shown in Figure 2.6 [34].

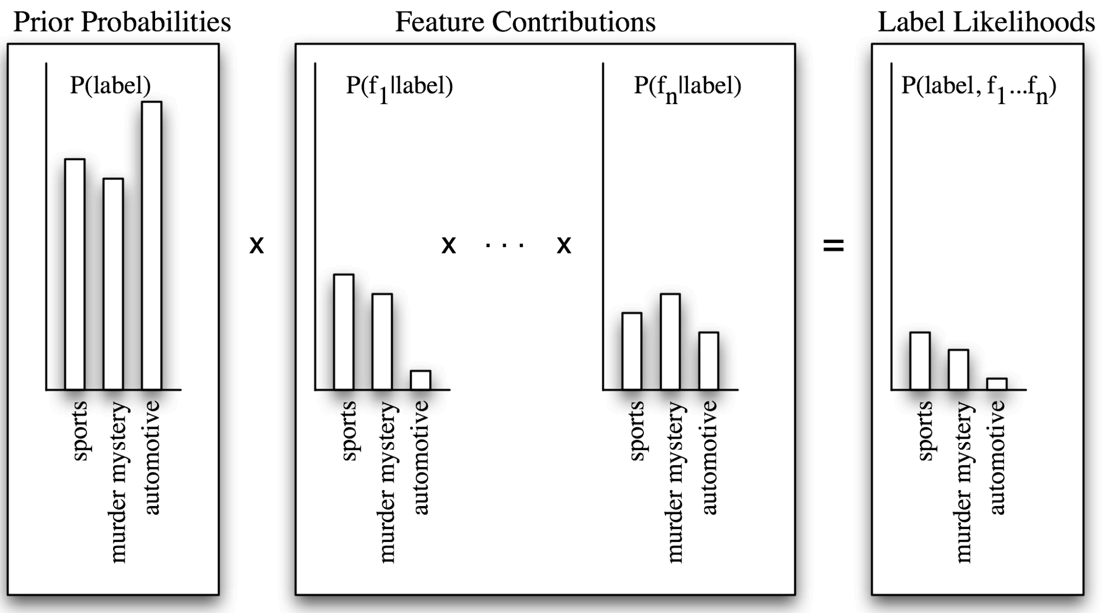

Figure 2.6: Calculating label likelihood in Naive Bayes to determine the right label

In Figure 2.6 it describes an example, where text is classified according to three labels prided sports, murder mystery and automotive, first the prior probability for each label is calculated, then feature contribution for each feature given each label is calculated, so for instance if feature <run> is present in 30 % of sports text, 25% in murder mystery and only 5 % in automotive, multiplying each contribution with the prior probability will result in likelihood, and the labels with highest likelihood is the right answer, which in this case is the sports label [34].

The Naive Bayes classifier is the most basic Bayes network, where each class has feature children as shown in Figure 2.7, and these children are independent, having no connections between them, and this is called conditional independency [36].

The reason Naive Bayes classifier is called Naïve, is because it is naive to assume that all features are independent, since that is not the case with real life problems, in fact it is hard to find features without any degree of dependency between them [34].

2.10.2 Support Vector Machines

Support Vector Machines (SVM) is the newest supervised techniques, which revolves around the idea of a margin, which is the sides of a hyperplane, a hyperplane defines the

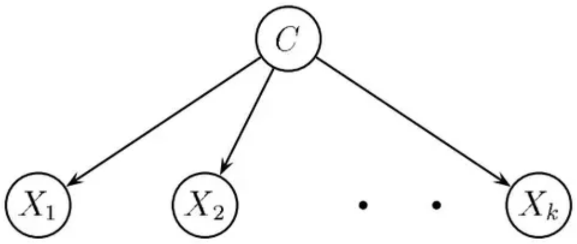

Figure 2.7: Conditional In-dependency of features in the same class

decision boundary of a classifier, it separates the data points into two classes, the points lying on each side belonging to a certain class, SVM aims to find optimal hyperplane that maximizes the margin, by finding the one with largest distance between hyperplane and instances of data on both sides, which has been proven to reduce the generalization error rate [13].

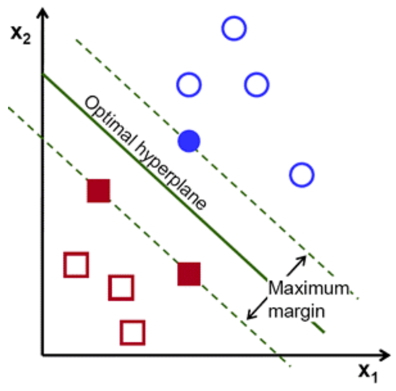

Figure 2.8: Finding optimal hyperplane using SVM

SVM has many advantages, some of which are, it's high dimensional space, it uses a small subset of training data for its decision function and it has different kernel functions, that can be specified to the decision function, the idea of kernel came up as a solution to non- linear separable data, as that is the case with most real world problems, where a hyperplane dividing two classes cannot be drawn, kernel transforms the data points to a higher dimensional space in which a hyperplane that separates data points could be found. There are many predefined kernels like linear, polynomial, Radial Basis Function (RBF), sigmoid as well as custom kernels [22].

2.10.3 Decision Trees

Decision Tree is a classifier that classifies data based on feature values of instances and sort them accordingly, it is one of the important supervised machine learning classifiers, since it can be applied to different real world problems. Each node in a decision tree is a feature of an instance to be classified, and each branch represent a possible value for it, Figure 2.9 shows a decision tree, where root node represent best feature to split data on, and branches represent values it can have, Table 2.3 shows the training data applied that resulted in tree in Figure 2.9. Finding the optimal decision tree lies in finding the feature that best split data, there are many ways to find such a feature, like information gain and gini index, there is no indication of a method being better than the other as it varies from one problem to another, therefore experimenting with different methods for a given problem might be relevant, this feature becomes root node and the same process is repeated for divided instances creating subtrees, until train data is divided into subsets of the same class [13].

Sample Feature Set				
at1	at2	at3	at4	Class
a1	a2	a3	a4	Yes
a1	a2	a3	b4	Yes
a1	b2	a3	a4	Yes
a1	b2	b3	b4	No
a1	c2	a3	a4	Yes
a1	c2	a3	b4	No
b1	b2	b3	b4	NO
c1	b2	b3	b4	No

Table 2.3: A sample of training data where each column represent a feature

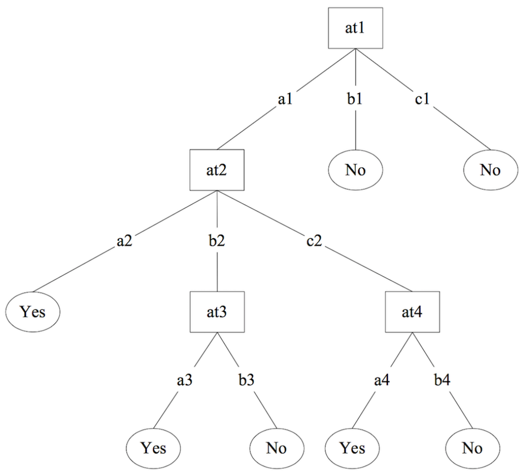

Figure 2.9: The Tree produced after training with the data in Table 2.3

2.11 Ensembles

Ensembles methods are algorithms that construct a set of classifiers, given an instance it classifies it according to a weighted vote of the classifiers [6].

2.11.1 Voting

The idea of ensembles originated from the thought that multiple experts opinion on a given problem is better than a single expert opinion, if they are properly combined, the same is with classifiers, there are different combination rules, simplest one is majority voting, if two or three classifier agree on a certain class then the result of the prediction is that class [12].

2.11.2 Random Forest

Random Forest is one of the most popular ensembles methods that yields great performance results, it is constructed from a set of decision trees, it is fast, robust and does not over fit, it utilizes decision trees to the fullest [27].

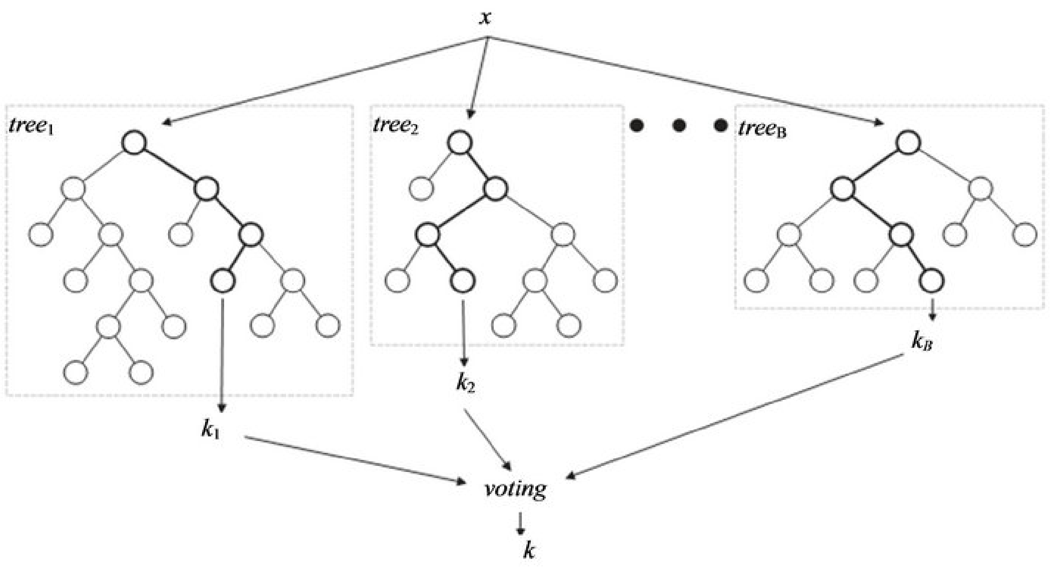

Figure 2.10: Illustration of a Random Forest

As shown in Figure 4.3,this Random Forest constructs B decision trees, given a feature x, with each tree outputting the label it thinks is the right answer, and if majority voting is used then the label with the most votes is chosen.

2.12 Word2Vec

It is an Unsupervised machine learning algorithm that is trained on a large text corpus, and outputs word vector for each word in the corpus's vocabulary without the aid of any external annotations. The notion of representing words in the form of word vectors has been around for some time with existing models to achieve that, but most of these model were computationally expensive, that is why when Tomáš Mikolov proposed Word2Vec, a model that was simple and not computationally expensive it became very popular.

Words in NLP tasks and techniques were treated as indices in a vocabulary, there was no relation between words,but with the Word2Vec model words are represented in terms of other words, as it uses distributed representation of words. Not only did the Word2Vec model imply similarity between words, it also made it possible to represent the relations between words in an algebraic notations, for instance vector(”King”)-vector(”Man”)+vector(”Woman”) results in a vector, which is closest to vector(”Queen”), as a King who is not a man but a woman, is a queen, a graph of these vectors can be seen in Figure 2.11. This form of representation attracted a lot of attention to Tomáš Mikolov Word2Vec's model [21].

Beside capturing the semantics of a word, word2vec is considered as a form of dimensionality reduction, instead of representing words in the from of one hot vectors, that are the size of vocabulary, which can became huge when, the vocabulary is large, Word2Vec can represent words in vectors with specified dimensions.

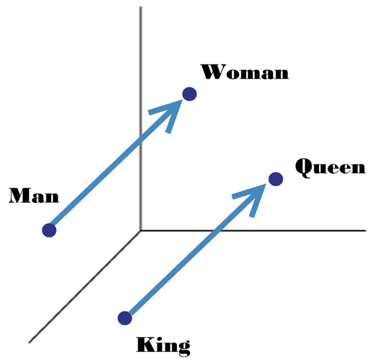

Figure 2.11: Plotting of word vectors of king, Man, Queen and Woman

Word2Vec is a shallow fully connected neural network, with a single hidden layer and output layer, with the number of nodes in the hidden layer being the number of dimensions of the desired vector. Word2Vec is based on the idea of a context, for example, if we have a sentence with a removed word, one would be able to guess the missing word, as the words surrounding the missing word, have something to do with it, as John Firth once said ”You shall know a word by the company it keeps” (Firth, J. R. 1957:11)”, thus a context of a word is the words surrounding this word or the neighbours of this words.

There are two proposed architecture for the word2vec model, The Continous Bag of Words (CBOW) and the Skip-Gram (SG), the CBOW model predicts a word given it's context, while SG predicts surrounding words given a word, both of these architecture have a window size which is used to represent the context of a word, if a window size 2 is chosen then the total number of words considered in context is 4, 2 words before and 2 words after.

2.12.1 Continuous Bag of Words

The Continuous Bag of Words model takes a window of size c words as a context, and predicts the most likely word associated with them. The Figure 2.12 shows the CBOW model neural architecture, where context words are represented as one hot vectors with a 1 in the position corresponding to the word in the vocabulary and the rest is zero, X stands for the context word, V stands for total number of vocabulary, N stands for number of nodes in hidden layer, while W_{VxN} is weight matrix between input layer and hidden layer, and W'_{NxV} is weight matrix between hidden layer and output layer, since input is a one hot vector, the product of the input and weight matrix results in selecting the row corresponding to 1, and since that we have multiple word vectors depending on the context which is c according to Figure 2.12, the activation function in hidden layer sums the product of each one hot vectors and Weight matrix which is the row in W_{VxN1} and dividing it by the c to get the average, then the weight matrix W'_{NxV} results in outputting a probability vector of size V, with each row presenting the probability of a word associating with the given context [28].

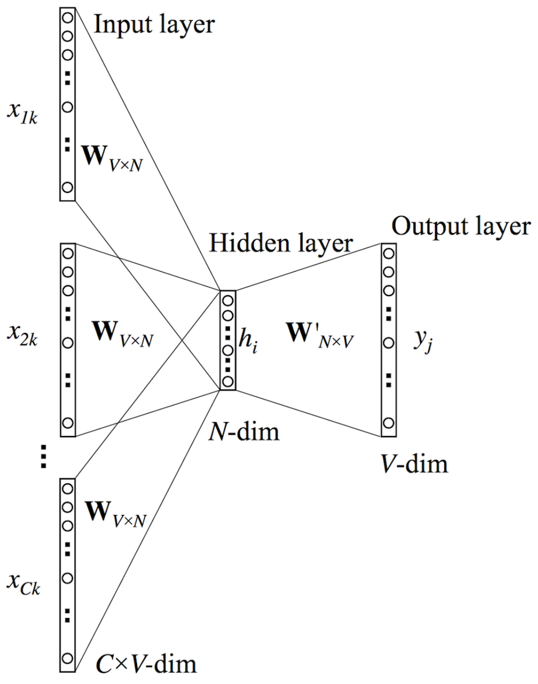

Figure 2.12: The Continuous Bag of Words architecture

2.12.2 Skip-Gram

Unlike the CBOW model, the Skip-Gram model takes a target word as an input and outputs context word vectors, Figure 2.13 shows the neural architecture of the model. The activation function for the hidden layer is simply copying the row corresponding to 1 in the weight matrix as we have seen before, but now instead of outputting one vector, C multinomial distributions vectors are outputted, with each vector representing a word in the context of the given target word [28].

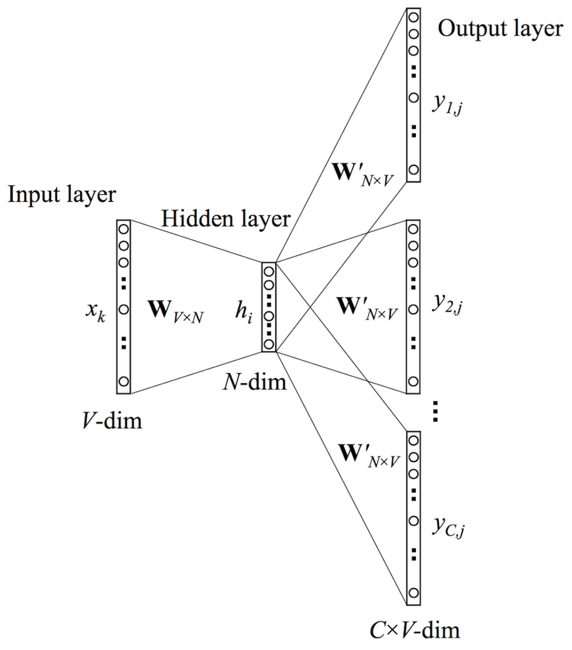

Figure 2.13: The Skip-Gram architecture

In the above mentioned models of word2vec, it can be noticed that dimensions of outputted word vectors is the same as that of input layer, which is the size of one hot vector, one could get confused since word2vec is said to reduce the dimensions of output words, but the fact is, these vectors are not the final output of the word2vec model.

Word2Vec model uses a techniques used a lot in machine learning, especially in unsupervised machine learning, it transforms an unsupervised problem to a supervised classification problem, the unsupervised problem in this case was given a set of words, find the corresponding word vectors with out any other information, word2vec changes this problem to either CBOW model where it uses context to find a target word, so target word is considered as a label, while in SG it uses a word to find context words that are considered labels in this task.

The goal of the supervised task is just to learn the weights of the hidden layer, after the model is trained, the final output is stripped of the output layer and the hidden layer is kept as the hidden layer controls the dimension of the resulting vector, this is called the decompression stage.

2.13 Software and Tools

The programming language used to implement the presented Arabic Part of speech tagger is Python 3.5 [35]. Python is a high level general purpose programming language, it is known for code readability and expressing ideas in fewer lines of code. Python was chosen as the programming language for this paper as it offers a great ecosystem with various libraries for scientific computation, that made it a popular language in the field of machine learning.

2.13.1 Scikit-Learn

Scikit-Learn is a Python module that implements a wide range of state of the art machine learning algorithms, this package mostly written in python makes use of the rich environment provided by Python, to deliver an easy to use interface to match the rising need for statistical analysis from non-experts in the software industry, and from fields outside of computer science such as biology, which made Scikit-Learn package as the best choice to use for machine learning [24].

2.13.2 Keras

Keras [4] is an open source neural network library written in Python, it is a higher level abstract library that can be run on deeplearing4j, Theano or Tensorflow, which are neural networks libraries as well, but using one of them for building neural network is not as straight forward and requires more knowledge in the inner workings of neural networks.

Keras has all the building blocks needed to build a neural network, from layers to activation and optimization functions, which makes it easier to implement a neural network abstractly regardless of the running background, therefore Keras was the right fit for implementing neural network in this paper.

2.13.3 Gensim

Gensim [26] is an open source vector space modeling library, which is written in Python, it is designed to handle large text data by using incremental algorithms, and that is what makes it different from other libraries. It has many implemented algorithms, the one most relevant to this paper is the word2vec algorithm, thus this library is used to model word2vec algorithm in the proposed Tagger.

Chapter 3

Methodology

This Chapter covers the NLP supervised classification process, and the implementation details of the proposed Arabic POS tagger.

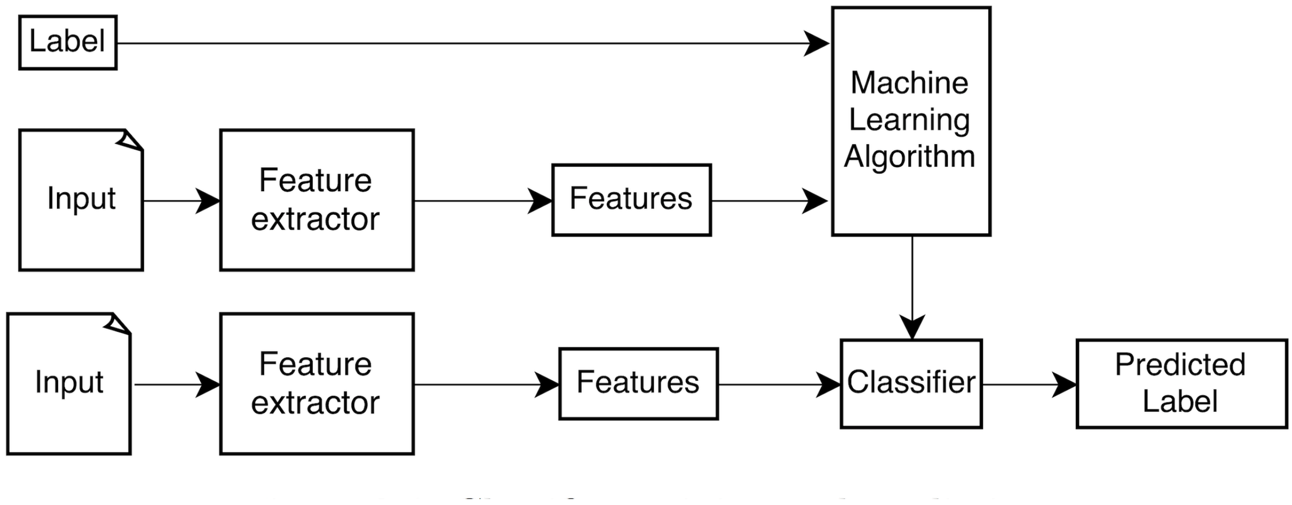

Figure 3.1: Supervised Classification

3.1 Supervised Classification

Supervised Classification is the task of assigning labels to a given input, during the training stage as described in Figure 3.1, the machine learning algorithm is given a set of labels and a set of features, produced after extracting features from raw input data, while during the prediction stage the classifier is given only the features set, and then it outputs the predicted label.

In this Chapter each stage of the classification process is going to be discussed.

3.2 Dataset

The data set used in training and testing the system, is the PADT 1.0, PADT is a collection of multi-level annotations based on MSA, that was developed to be used in general NLP tasks. It contains news texts and articles from Arabic GigaWord [23] also, it partly overlaps with PAT 1 and 2.

The PADT data is comprised of 6 text sources as shown in Table 3.1, with AFP, UMH and XIN not having a morphological files, as they were in the Early stages of constructing the dataset and morphological trees were not introduced yet, but morphological tags were present in the analytical files, while ALH, ANN and XIA had 3 files, a non-annotated, morphological and analytical [31].

The PADT data set			
Data Set	Tokens	Original Provider	Related Corpora
AFP	13,000	Agence France Presse	PAT Part 1
UMH	38,500	Al Hayat News Agency	PAT Part 2
XIN	13,500	Xinhua News Agency	Arabic Gigaword
ALH	5,000	Al Hayat News Agency	Arabic Gigaword
ANN	10,000	An Nahar News Agency	Arabic Gigaword
XIA	20,000	Xinhya News Agency	Arabic Gigaword

Table 3.1: This Table describes the data that comprises the PADT, and the number of tokens they contribute with.

3.2.1 Processing PADT

The aim of PADT processing is to get the input in the right shape to work with, and since PADT is essentially a collection of text, and text is a list of sentences, the input will be a list of tagged sentences, where each word in a sentence is paired with it's corresponding tag.

The PADT data set has multi-level annotations, analytical and morphological, focusing more on the later as analytical annotation focuses on the context of the words, the grammar, whether a word is a subject or an object, while morphology is concerned about structure of words and part of words.

In the morphological files provided by the dataset, words are represented in the form of morphoTrees, which are mapped to a tuple of values that are displayed in Figure 3.2, the most important element for our task is the second element, which is the tag [9].

As viewed in Figure 3.2, the tag is not as simple as expected, it is not just a Noun or a Verb, the tags shown consists of multiple tags, it can have up to 15 slots, with each slot adding to the tag, the first one being the basic POS tag, while the others are sub tags used to give more in depth knowledge about the word, if for instance, it is a Noun, whether it is singular or plural, feminine or masculine, and if it is a Verb, whether it is past or present, as shown in Table 3.2 [32].

String ··· Token	Token Tag	Buckwalter's Morph Tags	Token Form	Token Gloss
سيخبرهم	F---------	FUT	*sa-*	will
	VIIA-3MS--	IV3MS+IV+IVSUFF_MOOD:I	*yu-ḫbir-u*	he-notify
	S----3MP4-	IVSUFF_DO:3MS	*-hum*	them
بذلك	P---------	PREP	*bi-*	about/by
	SD----MS--	DEM_PRON_MS	*ḏālika*	that
عن	P---------	PREP	*ʿan*	by/about
طريق	N-------2R	NOUN+CASE_DEF_GEN	*ṭarīq-i*	way-of
الرسائل	N-------2D	DET+NOUN+CASE_DEF_GEN	*ar-rasāʾil-i*	the-messages
القصيرة	A-----FS2D	DET+ADJ+NSUFF_FEM_SG+ +CASE_DEF_GEN	*al-qaṣīr-at-i*	the-short
والإنترنت	C---------	CONJ	*wa-*	and
	Z-------2D	DET+NOUN_PROP+ +CASE_DEF_GEN	*al-ʾinternet-i*	the-internet
وغيرها	C---------	CONJ	*wa-*	and
	FN------2R	NEG_PART+CASE_DEF_GEN	*ġayr-i*	other/not-of
	S----3FS2-	POSS_PRON_3FS	*-hā*	them

Figure 3.2: Morphological tuples for words in PADT

Structure of PADT Tags				
Position	Previous Tag	Category	Value	Meaning
1	no condition	Part of speech	N	Noun
			V	Verb
			P	Particle
2	1:V	Primitive/Derived	O	Primitive
			A	Active Participle
			P	Passive Participle
			M	Masdar
			L	Location or Time
			T	Tools
			E	Elative
			C	Color
	1:N	Tense & Mood	P	Perfect
			I	Imperfect Indicative
			S	Imperfect Subjunctive
			J	Imperfect Jussive
			R	Imperative
			E	Energicus"-an"
			N	Energicus"-anna"

Table 3.2: Example for first and second slots in PADT tags.

3.2.2 Final Tagset

The PADT data set has 21 tags, these tags are too specific for the presented task, a much simpler basic tagset was needed, and thus they were mapped according to the Universal Tagset as shown in Table 3.3 . The Universal Tagset consist of 12 universal POS categories that are common between all languages, it is focused on POS tags that are expected to be most beneficial, to facilitate NLP tasks, since most data sets are language specific [25]. The above mapping resulted in the reduction of tagset from 21 to 11 tags, whose description is provided in Table 3.4, the dataset was further reduced from 11 universal tags to 8 tags, as the "X" tag, which stood for unknown words, "." tag, which was for punctuation, and "NUM" tag, which stood for numerical, were removed as shown in Table 3.4.

Universal Tagset										
ADJ	ADP	ADV	CONJ	NOUN	NUM	PRON	PRT	VERB	X	PUNCT
A-	P-	D-	C-	N-,Z-	Q-	S-,SD,SR	F-,FI,FN	VC,VI,Vp	-,I-,Y-	G-

Table 3.3: Mapping of PADT tags to Universal tags.

Universal Tagset	
Tag	Description
ADP (adpositon)	a term that covers prepositions and post-positions
CONJ(conjunction)	a word that links two or more words with each other by expressing a semantic relationship between them.
NOUN (noun)	a part of speech typical denoting a person,place,thing or animal.
PRON (pronoun)	pronouns are words that substitute nouns or noun phrases.
VERB (verb)	a word that typical signals events and actions.
PRT(particle)	are function words that mus be used with other words to impart meaning.
ADV (adverb)	are words that typically modifies verbs in categories such as time and place.
ADJ (adjective)	are words that typically describes nouns ,their properties and attributes.
NUM (numeral)	a word that expresses an number.
PUNCT (punctuation)	punctuation marks are non-alphabetical symbols used for delimitation.
X (unknown)	this tag is used for words that can not be assigned a part of speech tag.

Table 3.4: First column shows Universal Tags while second column offers description.

3.3 Feature Extraction

The process of Feature Extraction consists of first, constructing the features that are informative and distinctive, then using feature selection to select features that are most effective at differentiating, then finally representing the features.

3.3.1 Feature Construction

Data is represented by a fixed number of features, constructing these features is problem specific and related to available measurements, experts are often required for this stage, hence, why it is also known as feature engineering.

The stage of Feature Construction plays a crucial part in data analysis, it greatly affects the performance of the machine learning algorithm used, one should make sure not to discard any informative features, it is argued that it is best to err on being too inclusive rather than losing information [7].

3.3.2 Feature Selection

One might think that adding as many features as possible would increase the accuracy of the prediction, but that is not always the case as it results in either noisy features or the algorithm over fitting, and not being able to generalize when faced when a new example, hence, it is important to examine the effect of features on the model [18].

Feature selection revolves around finding the most informative, and relevant features of a feature set, primarily for improving the accuracy of prediction of the model, but it is also often used to better understand and visualize the data, as well as generally reduce the data, to free more space and increase algorithm speed [7].

There are many algorithms for automatic feature selection due to the presence of data with hundreds of variables, but the majority falls under three main methods, which are the following:

- Filter Methods
- Wrapper Methods
- Embedding Methods

The Filtering method is considered a prepossessing step as it ranks features, where highly ranked features are selected and applied to the predictor, while in Wrapper Method, the features selection criterion is based on the performance of features on a certain predictor, meaning that the predictor is wrapped by a searching algorithms that keeps selecting a subset of features, and the performance, and chooses the subsets that yields the highest. In Embedding methods, it includes variable selection as part of the training process without splitting data into training and testing sets [3].

The method applied in this Paper, is filter method, as mentioned before, filtering methods use feature ranking techniques to rank the features by ordering them, ranking methods are used due to their simplicity, they are used to find relevant features using statistics and probability. There are different ranking methods, but the ones tested on PADT resulting feature set, are Chi-Squared, which computes the Chi-Squared statistics between feature and label, and Mutual Information, which measures the dependency between two variables, it is zero if the two variables are independent, and higher values means higher dependency [3].

3.3.3 Feature Vectors

After deciding on the feature set, these features have to be represented in a numerical form, for them to be passed to a machine learning algorithm, so one hot encoding is used to represent each feature as a vector with a series of ones and zeros.

3.4 High Dimensionality Problem

After representing features in their vector form, a problem arose due to large number of variables introduced, since in one hot encoding, it considers the feature 'word':لعب and feature 'word':ذهب as two different features, so the minimum number of features is the same as the number of words in the dataset, which is around fifty thousands, and this results in a feature vector having high dimensionality, leading to large processing time and space inefficiency.

In this section two approaches to go around this problem will be discussed.

3.4.1 Dimensionality Reduction

Dimensionality reduction is projecting high dimensional data to a lower dimensional space, while preserving as much information as possible, these new reduced vectors may be used as features or simply as a mean for data visualization [7].

There are many approaches to Dimensionality Reduction, which are listed below:

- Feature selection
- Linear or Non-linear
- Supervised or Unsupervised

As previously mentioned, feature selection produces a subset of the original features, thus with less features the dimensions are reduced, using linear or non-linear approaches, a linear or non-linear mapping is used to extract new features, in supervised approach, it takes into account any information related to class, while in unsupervised it does not [5].

In this paper a linear approach was used, to reduce the feature vector, to visualize the data using truncated Singular Value Decomposition (SVD). SVD is a factorization technique, that factors a matrix M into three matrices U,E,V. Truncated SVD is different than regular SVD, in regular SVD given a matrix of size NxN, it produces a matrix with N columns, while in truncated SVD, it produces a matrix with specified number of columns, and that is how the dimensionality is reduced [10].

3.4.2 Word Vectors

In this approach, the feature set will be discarded altogether, instead it will map a word to a vector with specified dimensions, producing word vectors.

Assuming we have a vocab consisting of only 5 words, {King,Queen ,Woman ,Princess,Man}, to represent theses words, without prior knowledge of word vectors, one hot encoding will be used and the resulting vectors will be [1,0,0,0,0], [0,1,0,0,0], [0,0,1,0,0], [0,0,0,1,0], [0,0,0,0,1], with this representation, there is no relationship between the words with each other other than equality. In a Word2Vec encoding with a specified 4 dimensions as described in Section 2.12, a distributed representation of words is used, the resulting vectors are {[0.99,0.99,0.05,0.7], [0.99,0.05,0.93,0.6], [0.02,0.01,0.999,0.5], [0.98,0.02,0.94,0.1]}, with these dimensions each representing a certain property, Royalty, Masculinity, Femininity and age, respectively, now relationships between words could be inferred, these vectors abstractly represent the meaning of the word.

3.5 Implementation Details

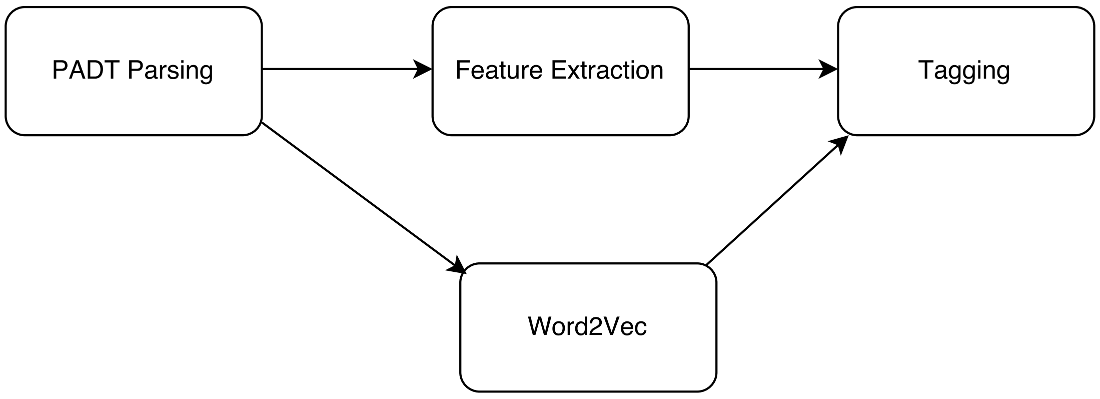

Figure 3.3: The structure of proposed Arabic Tagger

The implemented Arabic POS Tagger consists of four Python modules as displayed in Figure 4.1, the first one from the left, PADT Parsing is responsible for processing the PADT raw data and passing a list of tagged sentences to either Word2Vec or Feature Extraction modules, Feature Extraction is responsible for data visualization, by applying dimensionality reduction as well as feature extraction, from constructing the features to selecting them, and getting the data in the right shape to be passed to the classifier which is the tagging module.

The word2vec module is responsible for outputting a list of sentences where each word is a word vector and passing it to the classifier.

In this section implementation details of PADT Parsing, Feature Extraction and Word2Vec, are going to be discussed, while the Tagging module is going to be discussed in Chapter 4.

3.5.1 PADT Parsing

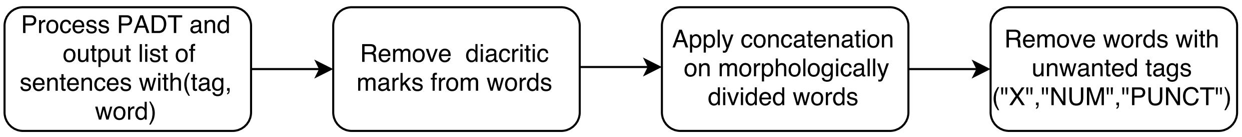
Process PADT and output list of sentences with(tag, word)
Remove diacritic marks from words
Apply concatenation on morphologically divided words
Remove words with unwanted tags ("X","NUM","PUNCT")

Figure 3.4: The process of parsing PADT and cleaning data

In this module as shown in Figure 3.4, the PADT data is processed and cleaned to be used in later stages, there are four stages in the module, which are discussed below.

In The first stage, the PADT dataset is read, and sentences consisting of pairs of word and it's corresponding tag are produced, as previously mentioned in Section 3.2, the PADT is composed of 6 subsets, where each text file, has a syntax file, which consists of tuples containing, a sentence number followed by, a number of tuples containing information regrading each word in the sentence, with the most relevant being the word itself, the POS tag and it's position in the given sentence.

Reading all syntactic files of the data set, with the above specified format,a temporary list consisting of sentences with tuples of the shape (tag,index,word) are formed, this temporary list sentence's tuples are ordered according to their index, which stands for their position in the sentence and removing the index variable altogether resulting in the desired tuple shape (tag,word).

Some of the words in the PADT dataset have Arabic diacritic marks, thus the second stage in this module is to remove these marks from such words, examples for removed diacritic marks, are subscript Alef and Shadda.

Since the PADT applies morphological analysis, words are striped from their prefixes and suffixes, for instance a word like "للمدرسة" is treated as two words, and split into "المدرسة" and "ل" with tags "NOUN" and "ADP" respectively, but that is not the expected behaviour for the proposed Tagger, as it expects it to be treated as a single word, as a result, a concatenation approach was developed to fix this problem.

In this approach simple rules were deduced and applied, tags of words which had the possibility of being a prefix or a suffix to other words like (”CONJ”,”ADP”,”PRT”,”PRON”) were examined, and by checking if they met the condition of being in this criteria, they were concatenated to the words after them if they were a prefix, or to the words before them if they were a suffix, some rules were added to the concatenation process, if the word started with "ال" and prefix was "ل" , the first character in "ال" was omitted, and if the word ended with "ى" , before adding a suffix it is replaced with "ي".

The last stage in this module is, filtering the tokens and removing un wanted ones, by going through their tags, for instance unknown words, punctuation symbols and numerical values, with tags ”X”,”PUNCT” and ”NUM” respectively.

The number of tokens after parsing the data set originally was 130,637 tokens, but after applying the suffix-prefix sentences fixes and contacting words, tokens were removed, thus the number of tokens was reduced to 123,687, then after removing tokens with unwanted tags, the number of tokens in the data set was significantly reduced to 42,631.

The final shape of the data is 8 tags and 42,632 tokens.

3.5.2 Feature Extraction

In this module Feature Extraction takes place, from constructing the features to encoding them, and either applying feature selection or dimensionality reduction as described in Figure 3.5.

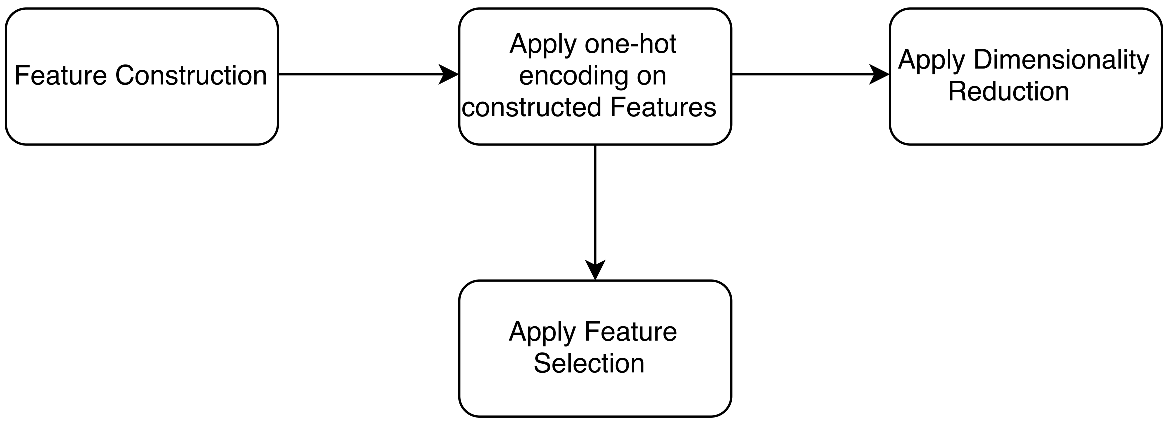
Feature Construction
Apply one-hot encoding on constructed Features
Apply Dimensionality Reduction
Apply Feature Selection

Figure 3.5: The process of Feature extraction applied to the PADT parsed data.

The first stage in this module is, Feature Construction, the features are engineered to fit the task at hand, which is tagging, then the set of chosen features as displayed in Table 3.5, are applied to each word in the PADT dataset.

Feature Construction	
Feature	Description
word	The word itself
word_length	The length of the word
is_first	True when the word is the first in a sentence
is_last	True when the word is the last in a sentence
prefix-1	The first letter in a word
prefix-2	The first two letters in a word
prefix-3	The first three letters in a word
suffix-1	The last letter in a word
suffix-2	The last two letters in a word
suffix-3	The last three letters in a word
prev_word	The previous word before this word
p_prev_word	The previous word to the word's prev_word
prev_tag	The previous tag before this word
p_prev_tag	The previous tag to prev_tag
next_word	The following word in the sentence to this word
n_next_word	The word in the sentences to the word's next_word

Table 3.5: First column shows the feature while second one offers description.

Feature selection is applied after applying one-hot encoding on the features, using Sickit-learn's SelectPercentile, which takes as an input the features and the percent of features which we want to select. Starting from 95% and decrementing the percentage gradually, while keeping track of the performance of the tagger, once it's accuracy is decreased we stop decrementing the percentage in SelectPercentile.

As previously mentioned in Section 3.3.2, the method used for feature selection is a linear method, which applies a ranking approach, there are two different ranking approaches used with SelectPercentile, chi-squared and mutual information classification. Feature selection is used to select a subset of features without losing information, so it could improve performance, if a suitable subset is found.

Dimensionality Reduction was first applied to visualize data as shown in Figure 3.6, the feature vectors were reduced to 2-D vectors using Sickit-learn's TruncatedSVD, which takes as an input the feature vectors, and the number of dimensions to be reduced to, and applies matrices factorization as explained in Section 3.4.1.

From Figure 3.6, each color represents a label, and in this case a POS tag, the data points are plotted according to their labels. It can be observed that some tags are sparse like the red one, and other tags are very common like the blue one, followed by the Violet one. It can also be observed that some tags are overlapping, and that is due to dimensionality reduction, since the feature vectors were of around fifty thousand dimensions but were reduced to only two.

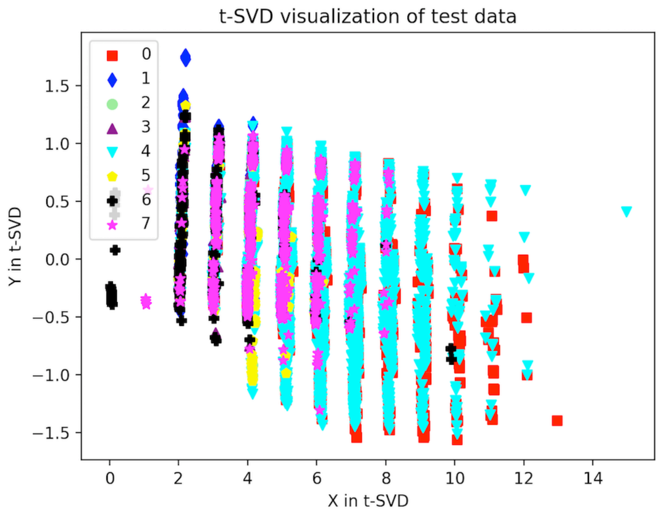

Figure 3.6: TruncatedSVD 2-D Reduction

3.5.3 Word2Vec

Gensim Python library is used for implementing a Word2Vec model, that is trained on the Arabic Gigaword dataset [23], The Figure 3.7 describes the word2vec module flow, in the next paragraphs each stage of the module will be discussed.

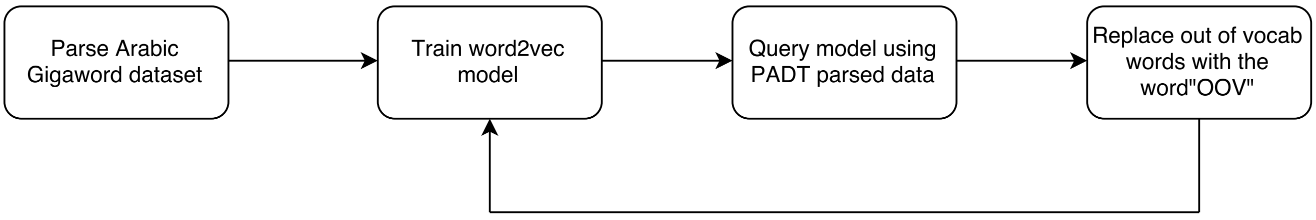
Parse Arabic Gigaword dataset
Train word2vec model
Query model using PADT parsed data
Replace out of vocab words with the word"OOV"

Figure 3.7: Training a Word2Vec model and using it to get the PADT parsed data corresponding word vectors

Arabic GigaWord fourth edition dataset was used to train the model, it is a comprehensive archive of MSA news wire, with 848,469 tokens, it consists of information resources from multiple news agencies as showed in Table 3.6. Processing this data set, numbers, punctuation characters and English letters were ignored, thus the number of tokens was reduced to 734,610, then the data was formatted to a list of sentences.

Training the model was very simple using Gensim library, the Gensim Word2Vec model uses the Skip-Gram architecture explained in Section 2.12.2 as default, the model had just to be provided with a list of sentences, which was prepared in the parsing stage, the number of dimensions of the desired word vector which is 300, and a context window of size 10, after training, the model is saved as to prevent training it at each run, as this process is time consuming.

Source	Tokens
Asharq Al-Awsat (aaw_arb)	36,694
Agence France Presse (afp_arb)	184,631
Al-Ahram (ahr_arb)	42,265
Assabah (asb_arb)	14,322
Al Hayat (hyt_arb)	209,318
An Nahar (nhr_arb)	253,559
Al-Quds Al-Arabi (qds_arb)	18,996
Ummah Press (umh_arb)	2,995
Xinhua News Agency (xin_arb)	85,689

Table 3.6: News sources for Arabic Gigaword data set and the number of token each contribute with.

Using the processed data, after parsing the PADT dataset, the trained model is queried to find corresponding word vector for each word in it, during this process a number of words from the PADT dataset were not found in the model's vocabulary, 118 words to be exact.

Instead of just ignoring the out of vocabulary words, these words were replaced with an <OOV> word, and the model was retrained with the sentences containing these words, by doing so adding a new word <OOV> to the model's vocabulary and having a vector representation for these words.

Chapter 4

Experiments and Evaluation

In this Chapter we are going to discuss the core module of the proposed Arabic Tagger, where the actual tagging takes place, using different machine learning algorithms, and presenting their results.

4.1 Tagging

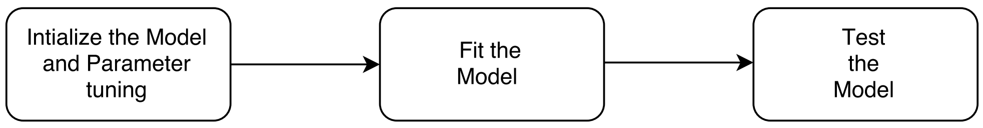

Figure 4.1: The Tagging module, where a machine learning model is initialized, fitted and tested.

As previously mentioned in Section 2.2, tagging is a classification task performed by a classifier. Before classifying, the present data is divided for training and testing.

As we can see in Figure 3.1, the process of classification is divided into two parts training and prediction, it is clear from the figure that the prediction stage is nearly the same as the training stage, the difference is that, in the training stage we offer the labels as inputs to the classifier, while in the prediction stage we expect the classifier to be able to output the right labels for the given input.

The Tagging module has three stages as seen in Figure 4.1, the first is to initialize the model and tune it's parameters, the next stage is fitting the model and this is the training stage, and the last stage is testing the model, which is the prediction stage.

4.2 Initializing the Model and Parameter Tuning

In this stage the machine learning algorithm that is going to be used is chosen, as there are various proposed algorithms in this paper, then parameter tuning is applied.

Parameter tuning, which is known as Hyperparaemter Optimization, is the task of finding the best parameters for a machine learning model, given a specific problem. These parameters improve the overall performance of the model.

Finding the best parameters is considered as a search problem, it could be done through brute force, by trying different parameters and comparing the results, Scikit-Learn offers a method that does all the work, the GridSearchCV method, which takes as an input a model, and a set of values for a given parameter, it tries all these values and compares the results of each one, and finally returns the best performing value for the given parameter and model.

Each model has it's own unique hyperparameters, that contribute to it's performance, in upcoming Subsections, hyperparameters of each classifier are going to be discussed.

4.2.1 Support Vector Machines

To achieve the best performing SVM model, first it is decided weather to use a linear SVM, that draws the hyper plane as a straight line separating two classes, or use a kernel SVM, that changes SVM's hyper plane to a non-linear one, then comes choosing the kernel method for non-linear SVMs, as there are various ones, RBF, sigmiod and poly, as stated in Section 2.10.2 ,and lastly tune the parameters for each kernel.

List of tuned Support Vector parameters:

- C: penalty parameter of the err.
- gamma: kernel coefficient for RBF, poly and sigmoid.

After supplying Scikit-learn's GridsearchCV, with the possible inputs for C and gamma, the best resulting values were C=1 and gamma=1000.

4.2.2 Random Forest

As previously mentioned in Section 2.11, Random Forest is an ensemble of decision tree classifiers, where each decision tree is split on suitable feature then voting ensues. Random Forest has many hyper parameters, the tuned Parameters are:

- criterion: this is the function that measures the quality of a split in a decision tree, possible function values are gini and entropy.
- max_features: is the maximum number of features a tree is built on, there are several values for it, none, which takes all features into account, sqrt, which takes square root of all features and 0.2, which takes 20% of features only.

- n_estimators: is the number of trees that are built before taking maximum or average voting.
- min_sample_leaf: is the number of leaves the end node should have in a decision tree.

After tuning the parameters, the resulting Random Forest classifier had the criterion entropy with 50 trees, all features were considered and a minimum sample leaf of 3.

4.2.3 Neural Networks

Choosing the suitable architecture for an artificial neural network can be challenging, as it has many parameters. There are two suggested neural architectures for this tagger, a Multi-Layer Perceptron (MLP) architecture and a RNN architecture.

The tuned parameters of neural networks are :

- number of hidden layers
- number of nodes in each layer
- activation functions
- method of initializing weight matrix
- loss or optimization functions

The Parameters for Neural networks were hand tuned, starting with one parameter and trying different values, then finally choosing the value which yields best results.

In both neural models presented, the input layer has a number of nodes equal to that of the input, and output nodes equal to that of output, since we have 8 tags, the output layer has 8 nodes, as for the input layer considering we have word vectors of dimensions 300, it has 300 nodes.

After Parameter tuning, the final MLP architecture is a shallow network with only one hidden layer, as adding further layers showed no improved performance, with the number of nodes in hidden layer being 64, the only layers having an activation function, were input and output layer, and the chosen function was softmax, which takes as an input a vector and outputs a vector of the same size, but with all the elements in it summing to a one. The method used for initializing was uniform method, which generates a uniformly distributed weight matrix, finally the used optimizer is adam, which is the loss function used to determine the learning rate of the model.

There are two sub-architectures based on RNN's model, LSTM and seq2seq LSTM, both of these architectures have the same basic parameters as MLP, adam as an optimizer, softmax as an activation function, and uniform weight initialization. They both consist of one input layer and one output layer and two hidden layers. In LSTM the input layer shape is (1,300), as it takes as an input a single word vector, while in seq2seq LSTM the input shape is (None,300), and that is because it takes as an input a sentence, which is a list of words, and it is None, because there is no fixed length for sentences as it varies. The output layer shape is (X,8), with X being, in case of LSTM only 1, as it outputs a single tag, and None in case of seq2seq, as it outputs a list of tags corresponding to a list of words, that it it took as an input. There are two hidden layers, the first consists of 64 nodes, while the second consists of 32 nodes.

4.3 Fitting The model

Fitting the model is the training stage, in which the input data together with it's labels are fed to the tagger, so it can learn. As previously mentioned in Section 4.1 the data is split for training and testing, 75% is used for training and the remaining 25% is used for testing.

4.4 Testing and Results

This module is responsible for performing the classification task, which is classifying the words, by predicting the correct label for each given input.

In this Section, we are going to experiment using multiple classifiers such as Naïve Bayes, SVM, Random Forest and neural networks for the proposed tagger and comparing their performance.

The performances of the classifiers is measured using accuracy, which is based on the overall success rate [14], by measuring the model's ability to correctly predict class labels, outputting an accuracy, which is the percentage of correctly classified instances in testing set.

4.4.1 Baseline Tagger

In this tagger, the input is passed to it after applying feature construction, using features described in Section 3.5.2, that are represented in the form of one-hot feature vectors, of 58,200 dimensions .

From Table 4.1, it can observed that when using this tagger, the highest performing classifier is Linear SVM yielding 95.72%, with Poly coming up next, followed by LSTM with close results, 94.67%, 94.42%, respectively. Random Forest performed fairly well, giving a 92.68% accuracy. Naïve Bayes, RBF SVM, Sigmoid SVM and MLP resulted in accuracies of 85.50%, 58.50%, 48.66% and 91.36%, respectively.

Classifier	Accuracy (%)
Naïve Bayes	85.50
Linear SVM	95.72
RBF SVM	58.50
Sigmoid SVM	48.66
Poly SVM	94.67
Random Forest	92.68
MLP	91.36
LSTM	94.42

Table 4.1: First column shows Classifier and second column shows Accuracy, when input is represented as one-hot feature vectors.

4.4.2 Applying Feature Selection

To further improve the performance of Baseline tagger, feature selection was applied, using Scikit-Learn's SelectPercentile, which given the percentage, selects features accordingly, 90% was used, as decreasing the percentage further, lead to a decrease in the original performance of the classifier, and chi-squared was used as the function to determine best feature to keep.

From Table 4.2, some performances were improved slightly, such as Naïve Bayes with 0.22%,while SVM classifiers remained unchanged,a slight decrease of performance of remaining classifiers can be noticed, with Random Forest, MLP, LSTM yielding 92.63 91.28 and 94.21, respectively.

It can be concluded that feature selection was not successful, as most of performances remained unaffected, and if selection percentage is decreased, the performance does.

The best performing classifier is still Linear SVM with 95.72%

Classifier	Accuracy (%)
Naïve Bayes	85.72
Linear SVM	95.72
RBF SVM	58.50
Sigmoid SVM	48.66
Poly SVM	94.67
Random Forest	92.63
MLP	91.28
LSTM	94.21

Table 4.2: First column shows Classifier and second column shows Accuracy, when 90% of features are selected using chi-squared.

4.4.3 Applying Dimensionality Reduction

Since the Baseline Tagger's feature vectors are of great dimensionality, around fifty thousand, this resulted in large processing time and memory inefficiency, thus Dimensionality Reduction was applied as explained in Section 3.4.1, and the feature vectors were reduced from 58,200 dimensions to only 500.

From Table 4.3 we could see some performance improving such as RBF SVM and MLP, both increased with 1.08%, with Sigmoid SVM remaining unchanged, while the performance of the rest of the classifiers decreased, with Naïve Bayes, Linear SVM, MLP and LSTM resulted in accuracies of 74.77%, 90.48%, 92.44% and 92.97%, respectively.The highest performing classifier is the LSTM.

Even though some of the performances were decreased, this experiment yielded good results, as the original feature vectors were reduced greatly, but still the classifiers performed fairly.

Classifier	Accuracy (%)
Naïve Bayes	74.77
Linear SVM	90.48
RBF SVM	59.58
Sigmoid SVM	48.66
Random Forest	92.08
MLP	92.44
LSTM	92.97

Table 4.3: First column shows Classifier and second column shows Accuracy, when Feature Vectors are reduced to 500 dimensions.

4.4.4 Using Word Vectors

Applying Dimensionality Reduction on the Baseline Tagger did not produce desired result, thus another approach was taken to reduce the input vectors.

In this approach, the feature vectors were ignored altogether and word vectors of only 300 dimensions were used, enabling classification based on the semantic relation of words only, with no prior feature extraction.

As observed from Table 4.4, the highest preforming classifier is the LSTM model with an accuracy of 95.76%, with MLP second to it with an 0.8% difference, while Naïve Bayes, Linear SVM, RBF SVM, Sigmoid SVM, Poly SVM, Random Forest and seq2seq LSTM resulted in accuarcies of 83.41%, 93.29%, 91.46%, 33.19%, 94.44% and 75.83%, respectively.

Classifier	Accuracy (%)
Naïve Bayes	83.41
Linear SVM	93.29
RBF SVM	91.46
Sigmoid SVM	33.19
Poly SVM	33.19
Random Forest	94.44
MLP	95.68
LSTM	95.76
seq2seq LSTM	75.83

Table 4.4: First column shows Classifier and second column shows Accuracy, when input is represented as Word Vectors of 300 dimensions.

4.4.5 Normalized Word Vectors

Normalizing word vectors showed improved performances across all classifiers, except Naïve Bayes and seq2seq SVM model, with the highest performing classifier being the MLP model with an accuracy of 96.29%, while Naïve Bayes, Linear SVM, RBF SVM, Sigmoid SVM, Poly SVM, Random Forest, LSTM and seq2seq LSTM resulted in accuarcies of 83.41%, 94.16%, 91.46%, 33.49%, 33.51%, 94.45%, 96.16% and 74.47%, respectively as shown in Table 4.5.

Classifier	Accuracy (%)
Naïve Bayes	83.41
Linear SVM	94.16
RBF SVM	91.46
Sigmoid SVM	33.49
Poly SVM	33.51
Random Forest	94.45
MLP	96.29
LSTM	96.16
seq2seq LSTM	74.47

Table 4.5: First column shows Classifier and second column shows Accuracy, when Word Vectors are Normalized

4.4.6 Adding Word Vectors to Feature Vectors

Instead of neglecting the constructed features altogether, and using only word vectors, word vectors were added for every word feature in the constructed features 3.5.2, producing the Combined Tagger.

This showed slight improvement of performances across many classifiers as displayed in Table 4.6, with the highest performing classifier being the LSTM, with an accuracy of 96.68% and an increase of 2.26%, while Naïve Bayes,Linear SVM, Random Forest and MLP resulted in the increase in accuracies by 0.25%, 0.53%, 2.63% and 2.32%. respectively.while performance of RBF SVM remained unchanged, that of Sigmoid SVM decreased slightly.

Classifier	Accuracy (%)
Naïve Bayes	85.75
Linear SVM	96.25
RBF SVM	58.50
Sigmoid SVM	34.43
Random Forest	95.31
MLP	93.68
LSTM	96.68

Table 4.6: First column shows Classifier and second column shows Accuracy, when Word Vectors are added to Feature Vectors

4.4.7 Applying Dimensionality Reduction to Combined Tagger

Classifier	Accuracy (%)
Naïve Bayes	81.90
Linear SVM	93.41
RBF SVM	58.50
Sigmoid SVM	41.70
Poly SVM	78.10
Random Forest	85.85
MLP	95.81
LSTM	95.88

Table 4.7: First column shows Classifier and second column shows Accuracy, when the combined feature vectors is reduced to 500 dimensions

Adding word vectors to the Baseline Tagger's feature vectors will increase it's dimensionality, thus, the resulting feature vector of the combined tagger will have 59,700 dimensions, and will be reduced to a feature vector of 500 dimensions as done previously in Section 4.4.3.

As seen before, reducing the dimensions to 500 did not yield better results, but produced overall good results.

From Table 4.7, the highest performing classifier is the LSTM with an accuracy of 95.88%,while Naïve Bayes, Linear SVM, RBF SVM, Sigmoid SVM, Poly SVM,Random

Forest and MLP generated accuracies of 81.90%, 93.41%, 58.50%, 41.70%, 78.10%, 85.85% and 95.81%, respectively.

4.4.8 Using Word Vectors instead of Words in Baseline Tagger

An alternative approach to combine word vectors and feature vectors of Baseline tagger is to substitute word features with word vectors, instead of just adding the word vectors to feature vectors as done in Section 4.4.6.

This approach will result in the reduction of Baseline feature vectors from 58,200 to 9,370 dimensions.

From Table 4.8,it can be observed that several performances increase with the highest performing classifier being the LSTM with an accuracy of 96.75% ,more than it's performance in the Baseline tagger with 2.33%.

Naïve Bayes, Linear SVM, RBF SVM, Sigmoid SVM, Random Forest and MLP generated accuracies of 85.30%, 95.87%, 58.50%, 34.50%, 95.44% and 95.64%, respectively.

Classifier	Accuracy (%)
Naïve Bayes	85.30
Linear SVM	95.87
RBF SVM	58.50
Sigmoid SVM	34.50
Random Forest	95.44
MLP	95.64
LSTM	96.75

Table 4.8: First column shows Classifier and second column shows Accuracy, when Word Vectors are added to Feature Vectors

4.5 Results Discussion

From experimenting with different input representations, it is observed that it affected the accuracy of classifiers, either increasing the performance or deceasing it.Table 4.9 shows each tagger and the experiment that made the tagger achieve it's highest accuracy.

It can be observed from table 4.9, that the overall highest performing classifier is the LSTM with an accuracy of 96.75%, when string words in features were substituted by word vectors instead and in the process reducing the dimensions.

The Linear SVM and acMLP are the next best performing classifiers yielding accuracies of 96.25% when applying the experiment in Section 4.4.7 and 96.23% when applying the experiment in Section 4.4.2.

Classifier	Accuracy (%)	Experiment
Naïve Bayes	85.75	Adding Word Vectors to Feature Vectors
Linear SVM	96.25	Adding Word Vectors to Feature Vectors
RBF SVM	91.46	Using Word Vectors
Sigmoid SVM	48.66	Baseline Tagger
Poly SVM	94.46	Baseline Tagger
Random Forest	95.44	Using Word Vectors instead of words in Feature Vector
MLP	96.23	Using Normalized Word Vectors
LSTM	96.75	Using Word Vectors instead of words in Feature Vector
seq2seq LSTM	75.83	Using Word Vectors

Table 4.9: First column shows Classifier, second column shows Accuracy and third column shows experiment

Naïve Bayes, RBF SVM, Sigmoid SVM,Poly, Random Forest and seq2seq LSTM generated accuracies of 85.75%, 91.46%, 48.66%,94.46, 95.44% and 75.83% respectively each when applying the following experiments mention in Sections 4.4.6, 4.4.4, 4.4.1, 4.4.1 and 4.4.6 respectively.

Chapter 5

Conclusion

The aim of this thesis was to build a supervised Arabic Part of Speech Tagger, with state of the Art results, experimenting with various machine learning classifiers such as Naïve Bayes, SVM, Random Forest and Neural Networks, and techniques such as Feature Selection and Dimensionality Reduction, to find the best performing classifier.

It is concluded that the overall highest performing classifier, in all the carried out experiments, is LSTM, with an accuracy of 96.75%, with Linear SVM coming up close with 96.25%, as well as MLP with an accuracy of 96.23%, therefore, LSTM outperformed all the other classifiers producing state of the art results, while Linear SVM and MLP yielded the same accuracy of 96.2%.

It is also notable to mention the performance of the seq2seq LSTM classifier, even though it yielded an accuracy of 75.8%, when using word vectors, it was interesting, since it is the only classifier that takes as an input a sentence, which is a sequence of words and outputs a sequence of tags, unlike other classifiers, which take as an input a single word and outputs a tag.

Chapter 6

Limitations and Future Work

6.1 Limitations

Although in this paper, the Arabic Tagger achieved results, that meets the research expectations, it is not clear of limitations, due to lack of time, delving into each technique in depth was almost impossible, this resulted in a non-deep understating of underlying mathematical concepts of presented machine learning algorithms and techniques, which affected hyperparameter optimization process, since it requires fair knowledge of parameters to fully manipulate them.

After processing the PADT dataset, the actual number of tokens used was nearly 40,000, which is a relatively small number for training, having a larger data set would have further improved the results, also to bypass the morphological properties of the dataset, words were concatenated together, which might introduce unknown errors, since it was hand crafted.

During experimenting, Poly SVM classifier was exempted from some experiments as it required a lot of time fro running and nearly ran forever, also the seq2seq LSTM classifier was only experimented with using word vectors, since it requires the input to be a sequence of sentences ,while Scikit-learn features are represented in a sequence of words, this affected the overall paper results, as it might affect the comparison between the classifiers.

The dataset was split into 75% for training and 25% for testing, evaluation of the Tagger was based only on prediction scoring of testing set, evaluation should be more through using other techniques, like cross validation.

6.2 Future Work

Improving the presented Arabic Tagger by handling the above mentioned limitations, as well as experimenting with new machine learning algorithms, and different learning paradigms, like unsupervised learning.

In this paper, the word vectors used are of 300 dimensions, trying different dimensions and comparing the results would be interesting, as it might produce better results.

Experimenting more with the sequence to sequence neural architecture, to further try and improve the results of seq2seq LSTM, as well as further manipulation of RNNs in general.

Appendix

Appendix

Lists

NLP	Natural Language Processing
POS	Part of Speech
PAT	Penn Arabic Treebank
HMM	Hidden Markov Model
MSA	Modern Standard Arabic
APT	Automatic Arabic POS-Tagger
AMT	Arabic Morpho-syntactic Tagger
ANN	Artificial Neural Network
RNN	Recurrent Neural Networks
SVM	Support Vector Machines
LSTM	Long Short Term Memory
seq2seq	sequence to sequence
RBF	Radial Basis Function
CBOW	Continous Bag of Words
SG	Skip-Gram
PADT	Prague Arabic Dependency Treebank
SVD	Singular Value Decomposition
MLP	Multi-Layer Perceptron

List of Figures

List of Tables

Bibliography

[1] N. Ababou and A. Mazroui. A hybrid arabic pos tagging for simple and compound morphosyntactic tags. *International Journal of Speech Technology*, 19(2):289–302, 2016.

[2] Rabab Ali Abumalloh, Hassan Maudi Al-Sarhan, Othman Ibrahim, and Waheeb Abu-Ulbeh. Arabic part-of-speech tagging. *Journal of Soft Computing and Decision Support Systems*, 3(2):45–52, 2016.

[3] Girish Chandrashekar and Ferat Sahin. A survey on feature selection methods. *Computers & Electrical Engineering*, 40(1):16–28, 2014.

[4] François Chollet. Keras, 2015.

[5] Manoranjan Dash, Hua Liu, and Jun Yao. Dimensionality reduction of unsupervised data. In *Tools with Artificial Intelligence, 1997. Proceedings., Ninth IEEE International Conference on*, pages 532–539. IEEE, 1997.

[6] Thomas.G Dietterich. Ensemble methods in machine learning. In *International workshop on multiple classifier systems*, pages 1–15. Springer, 2000.

[7] Isabelle Guyon and André Elisseeff. An introduction to feature extraction. In *Feature extraction*, pages 1–25. Springer, 2006.

[8] Nizar.Y Habash. Introduction to arabic natural language processing. *Synthesis Lectures on Human Language Technologies*, 3(1):1–187, 2010.

[9] Jan Hajic, Otakar Smrz, Tim Buckwalter, and Hubert Jin. Feature-based tagger of approximations of functional arabic morphology. In *Proceedings of the Workshop on Treebanks and Linguistic Theories (TLT), Barcelona, Spain*, 2005.

[10] Trent Hauck. *scikit-learn Cookbook*. Packt Publishing Ltd, 2014.

[11] Daniel Jurafsky and H. James. Speech and language processing an introduction to natural language processing, computational linguistics, and speech. 2000.

[12] Vandana Korde and C.Namrata Mahender. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, 3(2):85, 2012.

[13] Sotiris.B Kotsiantis, I Zaharakis, and P. Pintelas. Supervised machine learning: A review of classification techniques, 2007.

[14] Vincent Labatut and Hocine Cherifi. Accuracy measures for the comparison of classifiers. *arXiv preprint arXiv:1207.3790*, 2012.

[15] Aristomenis.S Lampropoulos and George. A Tsihrintzis. *Machine Learning Paradigms*. Springer, 2015.

[16] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015.

[17] Geoffrey Leech. Corpora and theories of linguistic performance. *Directions in corpus linguistics*, pages 105–122, 1992.

[18] David.D Lewis. Feature selection and feature extraction for text categorization. In *Proceedings of the workshop on Speech and Natural Language*, pages 212–217. Association for Computational Linguistics, 1992.

[19] Warren S McCulloch and Walter Pitts. A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133, 1943.

[20] Donald Michie, David.J Spiegelhalter, and Charles.C Taylor. Machine learning, neural and statistical classification. 1994.

[21] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[22] Iqbal Muhammad and Zhu Yan. Supervised machine learning approaches: A survey. *ICTACT Journal on Soft Computing*, 5(3), 2015.

[23] Robert Parker, David Graff, Ke Chen, Junbo Kong, and Kazuaki Maeda. Arabic gigaword, 2009.

[24] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(Oct):2825–2830, 2011.

[25] Slav Petrov, Dipanjan Das, and Ryan McDonald. A universal part-of-speech tagset. *arXiv preprint arXiv:1104.2086*, 2011.

[26] R. Řehřek and P. Sojka. Gensim–python framework for vector space mo delling. *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, 2011.

[27] Marko Robnik-Sikonja. Improving random forests. In *ECML*, volume 3201, pages 359–370, 2004.

[28] Xin Rong. word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*, 2014.

[29] Frank Rosenblatt. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[30] Vidushi Sharma, Sachin Rai, and Anurag Dev. A comprehensive study of artificial neural networks. *International Journal of Advanced research in computer science and software engineering*, 2(10), 2012.

[31] Otakar Smrz, Petr Pajas, Zdenek Zabokrtsky, Jan Hajic, Jiff Mirovsky, and Petr Nemec. Learning to use the prague arabic dependency treebank. *AMSTERDAM STUDIES IN THE THEORY AND HISTORY OF LINGUISTIC SCIENCE SERIES 4*, 289:77, 2007.

[32] Otakar Smrz, Jan Šnaidauf, and Petr Zemánek. Prague dependency treebank for arabic: Multi-level annotation of arabic corpus. In *Proc. of the Intern. Symposium on Processing of Arabic*, pages 147–155, 2002.

[33] Ilya Sutskever, Oriol Vinyals, and Quoc.V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[34] Richard.S Sutton and Andrew.G Barto. *Reinforcement learning: An introduction*, volume 1. MIT press Cambridge, 1998.

[35] Guido Van Rossum and Fred.L Drake.Jr. *Python tutorial.* Centrum voor Wiskunde en Informatica Amsterdam, The Netherlands, 1995.

[36] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.