Towards Automated Domain-Oriented Lexicon Construction and Dimension Reduction for Arabic Sentiment Analysis

Hasan A. Alshahrani
Western Michigan University, hasan_msh@hotmail.com

Follow this and additional works at: https://scholarworks.wmich.edu/dissertations

Part of the Programming Languages and Compilers Commons

Recommended Citation
https://scholarworks.wmich.edu/dissertations/3346
TOWARDS AUTOMATED DOMAIN-ORIENTED LEXICON CONSTRUCTION AND 
DIMENSION REDUCTION FOR ARABIC SENTIMENT ANALYSIS

by

Hasan A Alshahrani

A dissertation submitted to the Graduate College 
in partial fulfillment of the requirements 
for the degree of Doctor of Philosophy 
Computer Science 
Western Michigan University 
December 2018

Doctoral Committee:

Alvis Fong, Ph.D., Chair 
Elise de Doncker, Ph.D. 
Ikhlas Abdel-Qader, Ph.D.
ACKNOWLEDGMENTS

First, I would like to express my sincere gratitude to my advisor Dr. Alvis Fong for the support, motivation, the continuous teaching, and his guidance to finish this thesis. The best advisor I could have for my Ph.D study, my sincere thanks go to him.

Besides my advisor, I would like to thank the rest of my thesis committee: Dr. Elise de Doncker and Dr. Ikhlas Abdel-Qader, for their support and encouragement. I am thankful to them for the invaluable advices that shaped my work.

Last but not least, I acknowledge my family for their patience and sacrifice. My wife Fatimah, my daughter Sadeem, and my sons Tammam and Muhanned were the main source of my happiness and strength during my PhD journey.

Hasan A Alshahrani
Sentiment analysis is a type of text mining that uses Natural Language Processing (NLP) tools to identify and label opinionated text. There are two main approaches of sentiment analysis: lexicon-based, and statistical approach. In our research, we use the lexicon-based approach because the lexicon contains sentiment words and phrases which are the main linguistic units to express sentiments. More specifically, we work with domain-oriented lexicons as they are more efficient than general ones because the polarity is heavily driven by domains.

Arabic language has a degree of uniqueness that makes it hard to be processed with the available cross-language tools or use the direct translation from English. Arabic has 28 letters, and with the letters variations and vocalizations of letters, each letter might take up to 9 or more different shapes. Arabic is highly inflectional and morphological language, which makes it hard compared to English from features detection and dimension reduction perspectives. So, to get more accuracy, statistical learning methods have to be supported by language-specific knowledge.

In this research, we propose an approach called Polarity Latent Dirichlet Allocation (pLDA) to construct domain-oriented lexicon for an Arabic language domain. We first created our own training data, and we built our sentiment lexicon manually. After that, the
process was automated using the statistical model Latent Dirichlet Allocation (LDA), and then, the manual and automated results were compared. The two weaknesses of LDA were alleviated in our model by resolving the hyper-parameters problem and by enriching the corpora with more features of overall rated corpus. The lexicon was tested and validated for classification tasks with variety of data sets sizes, number of classes, and imbalance ratios. We designed rule-based fuzzy system especially to test our lexicon, and our approach showed excellent results as we got between 81% and 92% accuracy (according to text length and lexicon size).

Next step was a dimension reduction system for Arabic language. We developed a new stemmer for Arabic language and introduced it as an R package called arStemmer1. We compared our stemmer with the well known stemmer, Khoja stemmer which is one of the best performing stemmers. Our stemmer arStemmer1 outperformed Khoja in six out of seven experiments. We employed deep learning (skip-gram model) to build stop words lists with some manual filtration. The R package arStemmer1 is available for researches to use and test.
# TABLE OF CONTENTS

ACKNOWLEDGMENTS ................................................................. ii  
LIST OF TABLES ........................................................................ vi  
LIST OF FIGURES ...................................................................... vii  
LIST OF ABBREVIATIONS ......................................................... viii

1 INTRODUCTION AND BACKGROUND ....................................... 1 
   1.1 Overview .......................................................................... 1  
   1.2 Motivation ....................................................................... 3  
   1.3 Arabic Language Characteristics ...................................... 6  
       1.3.1 Transliteration ............................................................. 6  
       1.3.2 Letters Variations and Diacritics ............................... 11  
       1.3.3 Word Structure in Arabic (Morphology) ................. 13  
   1.4 The Main Goal and Contributions ..................................... 14

2 LITERATURE REVIEW .......................................................... 16 
   2.1 Sentiment Resources ...................................................... 16  
       2.1.1 English Lexicons ...................................................... 17  
       2.1.2 Arabic Lexicons ...................................................... 19  
   2.2 Stemmers ........................................................................ 24  
       2.2.1 Khoja Stemmer .......................................................... 25  
       2.2.2 ISRI Stemmer ............................................................ 26  
       2.2.3 Berkeley MT-Based Arabic Stemmer (BMTAS) ........ 28
Table of Contents—Continued

2.2.4 Berkeley Arabic Light Stemmer (BALS) ........................................... 28
2.2.5 TREC-2002-Enhancement ................................................................. 29
2.2.6 Light10 ......................................................................................... 31
2.2.7 Al-Stem ......................................................................................... 32
2.2.8 Sebawai Stemmer ........................................................................... 32

3 SENTIMENT ANALYSIS BASED FUZZY DECISION PLATFORM FOR STOCK MARKET ................................................................. 34

3.1 Introduction ........................................................................................ 34
3.2 Related Work ...................................................................................... 35
3.3 Data Collection: Corpora and Lexicon ................................................ 38
  3.3.1 Saudishares Corpus ...................................................................... 38
  3.3.2 Twitter Corpus .............................................................................. 39
3.4 Text Preprocessing and Features Reduction ........................................... 40
3.5 Saudi Stock Market Lexicon (SSML) .................................................... 41
3.6 Experiments ........................................................................................ 43
  3.6.1 Corpus-Based Approach (CBA) ..................................................... 43
  3.6.2 Semantical and Lexical Based Approach (SLBA) .......................... 44
3.7 Rule-based Fuzzy Decision .................................................................. 51
3.8 Conclusion .......................................................................................... 54

4 ARABIC DOMAIN-ORIENTED SENTIMENT LEXICON CONSTRUCTION USING LDA .......................................................... 55

4.1 Introduction ........................................................................................ 55
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>4.2</td>
<td>Related Work</td>
<td>58</td>
</tr>
<tr>
<td>4.3</td>
<td>LDA</td>
<td>61</td>
</tr>
<tr>
<td>4.4</td>
<td>Model Description</td>
<td>64</td>
</tr>
<tr>
<td>4.5</td>
<td>Documents Preprocessing and Aggregation</td>
<td>67</td>
</tr>
<tr>
<td>4.6</td>
<td>Experiment Results and Discussion</td>
<td>68</td>
</tr>
<tr>
<td>4.7</td>
<td>Conclusion</td>
<td>72</td>
</tr>
<tr>
<td>5</td>
<td>STEMMING</td>
<td>74</td>
</tr>
<tr>
<td>5.1</td>
<td>Introduction</td>
<td>74</td>
</tr>
<tr>
<td>5.2</td>
<td>Why Arabic Stemmer is Different ?</td>
<td>75</td>
</tr>
<tr>
<td>5.3</td>
<td>Related Work</td>
<td>76</td>
</tr>
<tr>
<td>5.4</td>
<td>Our Approach</td>
<td>77</td>
</tr>
<tr>
<td>5.4.1</td>
<td>Data Set</td>
<td>77</td>
</tr>
<tr>
<td>5.4.2</td>
<td>Lists Construction Using Deep Learning:</td>
<td>79</td>
</tr>
<tr>
<td>5.4.3</td>
<td>arStemmer1 Phases:</td>
<td>80</td>
</tr>
<tr>
<td>5.5</td>
<td>Experiment</td>
<td>84</td>
</tr>
<tr>
<td>5.6</td>
<td>R Package: arStemmer1</td>
<td>85</td>
</tr>
<tr>
<td>5.7</td>
<td>Conclusion</td>
<td>89</td>
</tr>
<tr>
<td>6</td>
<td>CONCLUSIONS AND FUTURE WORK</td>
<td>90</td>
</tr>
<tr>
<td>6.1</td>
<td>Summary</td>
<td>90</td>
</tr>
<tr>
<td>6.2</td>
<td>Future Work</td>
<td>92</td>
</tr>
<tr>
<td>REFERENCES</td>
<td></td>
<td>93</td>
</tr>
<tr>
<td>APPENDIX</td>
<td></td>
<td>101</td>
</tr>
</tbody>
</table>
LIST OF TABLES

1.1 Arabic-English Transliteration ........................................ 8
1.2 The Modified Transliterations Between Arabic and English ........ 9
1.3 Some Arabic Letters Variations and Diacritics ....................... 10
1.4 CV-template for the Arabic Word: Rasm ............................. 13

3.1 Number of Terms in the Saudi Stock Market Lexicon (SSML) .......... 42
3.2 Corpus-Based Classification Results Using SVM. ..................... 44
3.3 Accuracy of Statistical Learning Algorithms for Both Tweets and Forum Posts With Different Sizes. ......................... 46
3.4 Rule-Based Classification Performance for Data Set of Three Balanced Classes 48
3.5 Rule-Based Classification Performance for Data Set of Two Classes: Positive and Negative. ................................. 51

4.1 LDA-Lexicon of Tweets and Posts for Key Terms From 100 up to 10,000 66
4.2 Comparison Between LDA and Other Lexicons. ...................... 66

5.1 Prefixes and Suffixes Lists ......................................... 82
LIST OF FIGURES

1.1 Sentiment Analysis System ........................................... 3

2.1 Khoja Stemmer ............................................................... 27

2.2 Berkeley Arabic Light Stemmer (BALS) ................................. 30

2.3 TREC-2002 Stemmer Enhancement ...................................... 31

3.1 The Fuzzy Decision Main Components ................................. 53

4.1 pLDA Graphical Model. .................................................. 65

4.2 LDA Input and Output .................................................... 65

4.3 Accuracy of LDA Lexicon ................................................ 69

4.4 Sample of Terms for Each Polarity (Topic). .......................... 71

5.1 Larkey Stemmer ............................................................. 78

5.2 Skip-Gram Model ........................................................... 81

5.3 arStemmer Phases ......................................................... 82

5.4 Khoja Vs arStemmer1 ..................................................... 86

5.5 arStemmer1 Main Parts ................................................... 88

5.6 arStemmer1 Algorithm .................................................... 88
## LIST OF ABBREVIATIONS

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSML</td>
<td>Saudi Stock Market Lexicon</td>
</tr>
<tr>
<td>SA</td>
<td>Sentiment Analysis</td>
</tr>
<tr>
<td>MT</td>
<td>Machine Translation</td>
</tr>
<tr>
<td>OMCS</td>
<td>Open Mind Common Sense</td>
</tr>
<tr>
<td>AWN</td>
<td>Arabic WordNet</td>
</tr>
<tr>
<td>EWN</td>
<td>Euro WordNet</td>
</tr>
<tr>
<td>PWN</td>
<td>Princeton WordNet</td>
</tr>
<tr>
<td>MSA</td>
<td>Modern Standard Arabic</td>
</tr>
<tr>
<td>SAMA</td>
<td>Standard Arabic Morphological Analyzer</td>
</tr>
<tr>
<td>ESWN</td>
<td>English SentiWordNet</td>
</tr>
<tr>
<td>PATB</td>
<td>Penn Arabic Treebank</td>
</tr>
<tr>
<td>YT</td>
<td>YouTube Lexicon</td>
</tr>
<tr>
<td>GI</td>
<td>General Inquirer</td>
</tr>
<tr>
<td>Acronym</td>
<td>Description</td>
</tr>
<tr>
<td>---------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>PMI</td>
<td>Pointwise Mutual Information</td>
</tr>
<tr>
<td>SVM</td>
<td>Support Vector Machine</td>
</tr>
<tr>
<td>OM</td>
<td>Opinion Mining</td>
</tr>
<tr>
<td>NNLM</td>
<td>Neural Network Language Model</td>
</tr>
<tr>
<td>CBA</td>
<td>Corpus-Based Approach</td>
</tr>
<tr>
<td>SLBA</td>
<td>Semantical and Lexical Based Approach</td>
</tr>
<tr>
<td>PDA</td>
<td>Penalized Discriminant Analysis</td>
</tr>
<tr>
<td>KNN</td>
<td>K-Nearest Neighbors</td>
</tr>
<tr>
<td>BSS</td>
<td>Best Subset Selection</td>
</tr>
<tr>
<td>OVA</td>
<td>One-vs-All</td>
</tr>
<tr>
<td>NB</td>
<td>Naïve Bayes</td>
</tr>
<tr>
<td>MASC</td>
<td>Multi-domain Arabic Sentiment Corpus</td>
</tr>
<tr>
<td>HMM</td>
<td>Hidden Markov Model</td>
</tr>
</tbody>
</table>
CHAPTER 1
INTRODUCTION AND BACKGROUND

1.1 Overview

“Sentiment Analysis (SA) or Opinion Mining (OM) is the computational study of people’s opinions, attitudes and emotions toward an entity” [1]. Sentiments means the written emotions and not the mental state of the sentiment source. The sentiment is an opinion issued by a source towards a target. The target can be almost anything, a topic, a person, a product, or an organization [2].

The interest in sentiment analysis has been increasing in recent years [3]. The evolution of blogging and micro-blogging has increased this interest in sentiment analysis as it became the source of huge amounts of people reviews and comments about services, products, and so many other topics such as politics.

With the rapid growth of internet and social media, numerous amount of opinions are produced every day in the form of posts, messages, and tweets. This huge collection of text can be very valuable if studied very well and converted from unstructured to structured useful text. Some companies for example might spend money and time to extract the opinions of their customers about a product or a service but still cannot get the same number and quality of responses that are being posted on-line on daily basis.
The sentiment or the opinion has five main parts: target, source, aspect, the opinion itself and the time of the sentiment [3]. The source is also called the sentiment holder which means the person who said or wrote the sentiment. The target is the entity about which the sentiment/opinion is written. The aspect is a sub-attribute of the target, for example, if the target is a car, then the aspect might be the size or the speed of the car. The sentiment quintuple can be written as $N = \{G, A, S, H, T\}$, where $G$ is the target, $A$ is the aspect, $S$ is the sentiment, $H$ is the holder (source of sentiment), and $T$ is the time of the sentiment. The tasks of sentiment analysis are based on those five parts of sentiment [4]. The main categories of sentiments analysis tasks can be summarized as the identification and classification the five parts of the sentiment. In other point of view of the main task of sentiment analysis [5], it is all about emotion recognition and polarity detection [6] where polarity can be positive, negative, or neutral. Sometimes, both emotion recognition and polarity detection are considered as one task which is identifying the polarity (binary classification) using the emotions available in a document with assigning a degree for the polarity such as positive, very positive, and so on. Distinguishing between subjectivity and objectivity in a document can be advantageous for polarity classification [5]. Other tasks of sentiment analysis include but not limited to: aspect extraction, sarcasm detection, and personality recognition [7].

The main structure of sentiment analysis system can be summarized by Figure 1.1. The process can be done on three levels, sentence level, document level, and word (token) level. Sentiment analysis is divided into three main types: supervised, unsupervised and hybrid approaches. The supervised approach means to use a labeled corpus to train the any statistical algorithm/model such as support vector machine (SVM). The unsupervised means
to use a lexicon (dictionary) of sentiment words as a knowledge base to classify documents [8].

Figure 1.1: The Main Structure of Sentiment Analysis System

1.2 Motivation

Sentiment analysis has so many applications in different forms. Many companies adopted sentiment analysis as a daily task of their business process to measure the customer satisfaction as an example. In addition to that, sentiment analysis can enhance other systems
to do their job perfectly such as spam detection in social media, entertainment, customer relationship management, and recommendation systems [5]. Real time sentiment analysis tools have been getting more attention as they are doing crucial mission for both commercial and government intelligence applications [5]. Some examples of those tools are: SAS \textsuperscript{1}, IBM \textsuperscript{2}, and SenticNet \textsuperscript{3}.

The main reason we preferred to work with Arabic instead of adopting general approach that can be applied on any language, is because we believe that sentiment analysis methodologies and tools performance can be easily misguided by some linguistics tricks such as negation [5]. In other words, it is hard for statistical methods alone to do text classification with high accuracy as they are semantically weak [9]. From the resources (lexicons and datasets) perspective, which is the corner stone of sentiment analysis, a study done by [10] showed that it is better to build resources for the language under study (in-language) instead of translating data from other languages because machine translation (MT) itself is suffering the issue of resources availability more than sentiment analysis. One more reason not to use cross language sentiment analysis and stick with in-language sentiment analysis, is the divergence of sentiments from language to language [10]. The experiments done by [10] proved with evidence that sentiment analysis resource should be collected from the in-language instead of bringing it (translate it) from different language. So a source deprived language such as Arabic must be built from inside first not from outside to avoid any contamination resulted from inaccuracy of MT and cultural differences between languages.

\textsuperscript{1}https://www.sas.com/en_us/connect.html
\textsuperscript{2}https://www.ibm.com/analytics
\textsuperscript{3}http://business.sentic.net/
Arabic is a widely spoken language as it is the mother tongue of about 200 million native speakers nowadays from North Africa to the Middle East. In addition to Arabic speaking countries, all Islamic countries are highly influenced by the language-specific nature of Islam. Many Muslims believe that they have to learn Arabic to recite the holy Qur’an. Lexicons of indigenous languages of Islamic countries have borrowed some Arabic words either because of the Islamic influence during the Islamic empire time, or by learning Arabic language recently [11].

Arabic has a degree of uniqueness which makes it sometimes hard be processed with available cross-language tools (stemmers for example) or to use translated resources translated from English to Arabic. Arabic letters cannot be capitalized, can be elongated, and take different shapes according to the letter location in the word. Diacritics are special symbols written above and below the Arabic letters to give different pronunciation and meaning of words [12]. The word thahaba is very different from the word thahab, the first one means “he went”, and the second one means “gold”. Moreover, Arabic writing is from right to left and the word root in Arabic can be a seed for many words when affixes added to it [12].

From stemming perspective (stemming means reducing changed words to their original form or base, more about stemming in chapter 5), statistical methods have been doing good job in stemming as they can infer the common prefixes and suffixes by analyzing a text corpus. But they are still suffering from some limitations such as the problem of Arabic infixes and the uncommon suffixes and prefixes [13].
1.3 Arabic Language Characteristics

In this section, we will give a brief introduction about the Arabic language, and more details can be found in [14]. First step to get in contact with Arabic is alphabet transliteration, to be used in the rest of this thesis as discussed in the following section.

1.3.1 Transliteration

Transliteration is the mapping between two languages alphabets where a language is written using the others language’s alphabet supported by substitutions if needed (in case if one of the languages doesn’t have the same or close letter). Arabic alphabet is written from right to left and contains 28 letters. Each letter is written in different forms according to its position in the word: initial, medial, and final. The transliteration between English and Arabic is to romanize the Arabic letters into Latin letters by adopting a writing system or mapping between the two lists of alphabets with some substitutions and support by extra symbols as needed [15, 14]. As shown in Table 1.1, we built a transliteration system based on Habash et al. system [15], Buckwalter’s system [16], and Wikipedai 4. The letter name can be pronounced as shown in Table 1.1(from Wikipedia). We modified eight letters for two main reasons:

- To be closer to the Arabic shape as in the letters $\varepsilon \hat{\ell}$ and $\varepsilon \hat{\ell}'$

- To maintain the same level of similarity between adjacent Arabic letters in their peers in English such the two Arabic letters $\vartheta / \vartheta$, their transliterations are $t / ïˆ$, and the

\[4\text{https://en.wikipedia.org/wiki/Arabic_alphabet}\]
letters س/ش , s/š. In the letter š, we flipped the cursor symbol used by Habash et al. so it can be very similar to the three dots used in the Arabic letter ش. The same thing with the similar Arabic letters د/ذ, the should have peers with same or close level of similarity as we showed in Table 1.1: d/đ.

The modified Arabic letters are: ُ, ِ, َ, ﻓ, ْغ, ﻓ, ﺕ, ﺞ, ﺞ, ﺞ, ﺞ, and they were mapped respectively to the following symbols/letters: ′, £, £‘, ‡, š, a, Ĥ, đ, as shown in Table 1.2.
Table 1.1: Transliteration Between Arabic and English Letters

<table>
<thead>
<tr>
<th>Letter</th>
<th>Transliteration</th>
<th>Letter name</th>
<th>Example of pronunciation</th>
<th>form/location: Final-Medial-Initial</th>
</tr>
</thead>
<tbody>
<tr>
<td>ء</td>
<td>'</td>
<td>hamzah</td>
<td>apple</td>
<td>ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء ء Τ表</td>
</tr>
</tbody>
</table>
Table 1.2: The Modified Transliterations Between Arabic and English

<table>
<thead>
<tr>
<th>Letter</th>
<th>Transliteration</th>
<th>Letter name</th>
<th>Buckwalter</th>
<th>Habash</th>
</tr>
</thead>
<tbody>
<tr>
<td>ء</td>
<td>`</td>
<td>hamzah</td>
<td>'</td>
<td>'</td>
</tr>
<tr>
<td>غ</td>
<td>£</td>
<td>`ayn</td>
<td>E</td>
<td>ς</td>
</tr>
<tr>
<td>غ</td>
<td>£`</td>
<td>`ayn</td>
<td>g</td>
<td>γ</td>
</tr>
<tr>
<td>ث</td>
<td>‹</td>
<td>thā`</td>
<td>v</td>
<td>θ</td>
</tr>
<tr>
<td>ش</td>
<td>ſ</td>
<td>shīn</td>
<td>$</td>
<td>ˇs</td>
</tr>
<tr>
<td>ا</td>
<td>a</td>
<td>`alif</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>خ</td>
<td>Ḥ</td>
<td>khā`</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>ذ</td>
<td>đ</td>
<td>dhāl</td>
<td>*</td>
<td></td>
</tr>
<tr>
<td>Letter/diacritic</td>
<td>Letter name/definition</td>
<td>Transliteration</td>
<td>Pronunciation</td>
<td>Example</td>
</tr>
<tr>
<td>------------------</td>
<td>------------------------</td>
<td>----------------</td>
<td>---------------</td>
<td>---------</td>
</tr>
<tr>
<td>/sukun/</td>
<td>sukoon: the absence of vowel</td>
<td>*</td>
<td>no sound /silent</td>
<td>jabal'a</td>
</tr>
<tr>
<td>/shaddah/</td>
<td>shaddah : this mark means to double the consonant</td>
<td>~</td>
<td>vowel sound followed by consonant sound for the same letter, e.g. dda</td>
<td>Laddala/Lad¬¬¬¬¬ala</td>
</tr>
<tr>
<td>/fat-hah/</td>
<td>fat-hah: short /a/</td>
<td>a</td>
<td>/a/</td>
<td>dahaba</td>
</tr>
<tr>
<td>/kesrah/</td>
<td>kesrah: short /i/</td>
<td>i</td>
<td>/i/</td>
<td>laJiba</td>
</tr>
<tr>
<td>/dammah/</td>
<td>dammah: /u/ sound with joined lips</td>
<td>u</td>
<td>/u/</td>
<td>kutiba</td>
</tr>
<tr>
<td>/maddah/</td>
<td>maddah: used to lengthen the vowel, and it comes over &quot;alif(l), yaa'(ي), or waw(و)</td>
<td>~</td>
<td>e.g. with &quot;alif/Á/</td>
<td>Ám أيام</td>
</tr>
<tr>
<td>/tanween fatH/</td>
<td>tanween fatH</td>
<td>ã</td>
<td>/an/</td>
<td>a\m أمة</td>
</tr>
<tr>
<td>/tanween kas'r/</td>
<td>tanween kas'r</td>
<td>i</td>
<td>/in/</td>
<td>a\m أمة</td>
</tr>
<tr>
<td>/tanween Dham/</td>
<td>tanween Dham</td>
<td>õ</td>
<td>/un/</td>
<td>a\m أمة</td>
</tr>
<tr>
<td>/tA'marbutah/</td>
<td>tA'marbutah: always in the final position, make the noun feminine, and followed by nunation</td>
<td>/t/ , that</td>
<td>baJeedaun بعيدة</td>
<td>far (feminine)</td>
</tr>
<tr>
<td>/'alif mequraah/</td>
<td>'alif mequraah: it is another form of the letter 'alif(l), always appears in the final position of the word</td>
<td>Á</td>
<td>/ä/</td>
<td>'atÁ/ataa اتى</td>
</tr>
</tbody>
</table>
1.3.2 Letters Variations and Diacritics

There are some letters which have some other forms not mentioned in Table 1.1, and some symbols (diacritics) that play important role in the pronunciation chart of Arabic letters. Those characters are briefly explained in Table 1.3 [14, 17].

In addition to that, the hamzah letter can take some forms according to some rules [14]. The hamza can be placed either by itself (aloof) or sitting on one of the three chairs: ‘alif(ال), yā’ (ي), or wāw(و). The following are some examples for each:

- **hamzah on ‘alif**: hamza can be written above the ‘alif as in the first three examples, and can be below the ‘alif as in the last example: my father/‘Aby أبي, welcome/‘Ahlā اهلاء, command/‘Amr أمر, preparation/Istīf‘dad استعداد

- **hamzah on yā’**: the chair in this case is called yā’ but that is not accurate sometimes because the yā’ is not yā’ without its dots ي. Two examples are shown below, the first one is for hamzah on chair and the second one is for hamzah on yā’:
  
  fabulous/rA‘iF رائع

- **hamzah on wāw**: conference/mo’tamar مؤتمر, delayed/moA‘j‘al مؤجمه.

- **independent hamzah**: the hamzah in this case sits apart (aloof) such as: sky/sam‘اء سماء

  tA‘Marbutah

  tA‘marbutah is used with feminine gender. For example, the word beautiful in English refer to both feminine and masculine while in Arabic that is not the case. For the feminine gender, it is written as jameelah/جميلة, and for masculine (singular), it is written
as jameel/ جميل. From pronunciation perspective, tAʿmarbutah is sometimes pronounced as /t/ and sometimes as /h/ depending on the final inflectional vowel and the sound before it.

Shaddah

Shaddah in Arabic, represents consonant doubling process called tashdeed [14]. Shaddah is written above the consonant to indicate intensified pronunciation. Here are some examples: waDaHa/he clarified وَضَحَّ, kabʿara/enlarged كَبْرَا.

`alif Meqsuraah

`alif meqṣūrah ي is a form of the letter `alif ا and has the same shape of yA but without dots. It is always used at the end or words with almost all forms of words such as verb, noun, adjective, preposition, and proper names. Here are some examples ordered as verb, noun, preposition and proper name:AʿtA/he came هَدَى, hudA/guidance هَدَى, SuʿrA/the smallest (feminine) صَغْرَى, IIA(elaa)/to إِلَى.

Diacritics

Arabic diacritics (harakAt or taškeel in Arabic) are those little marks (short vowels) that are always associated to letters to avoid any mispronunciation. As we mentioned in Table 1.3, the Arabic diacritics are (shown on the letter s/س): fatHah، kasrah، Damrah، sukoon، šadʾah، tanween fatH، tanween kasr، tanween Dham، and madʾah which has the shape ∼. Examples are shown in table 1.3. Diacritics are optional except in some few cases such as educational texts and religious scriptures. This causes some difficulty
Table 1.4: CV-template

<table>
<thead>
<tr>
<th>Verbal noun</th>
<th>Masculine singular</th>
<th>Masculine singular</th>
<th>Feminine singular</th>
<th>Feminine singular (passive)</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>CvCC</strong></td>
<td><strong>CvCvCv</strong></td>
<td><strong>CvCvCv</strong></td>
<td><strong>CvCvCv</strong></td>
<td><strong>CvCvCvC</strong></td>
</tr>
<tr>
<td>draw</td>
<td>he has drawn</td>
<td>They have drawn</td>
<td>she has drawn</td>
<td>It was drawn</td>
</tr>
<tr>
<td><strong>rasm</strong></td>
<td><strong>rasama</strong></td>
<td><strong>rasamuo</strong></td>
<td><strong>rasamat</strong></td>
<td><strong>rusimat</strong></td>
</tr>
<tr>
<td>رسم</td>
<td>رسم</td>
<td>رسم</td>
<td>رسم</td>
<td>رسم</td>
</tr>
</tbody>
</table>

for automated text analysis as we will see in the following chapters of this thesis.

1.3.3 Word Structure in Arabic (Morphology)

Arabic word consists of two main morphemes (morphological parts): root and pattern [14]. The root is a morpheme that has invariable sequence of three to five consonants that can assimilate affixes to produce new words or forms. The pattern is the additional information added to the root such as tense. In other words, several forms of a word have the root in common and can be differentiated by patterns. Both pattern and root are discontinuous. We will use what is called CV-template [18] to represent root-pattern construction for the rest of this thesis (as needed) where C means consonant (root) and V means vowel (pattern). To make it clear, we gave an example about representing the word rasm/drawing رسم in Table 1.4.
1.4 The Main Goal and Contributions

The main goal of this thesis is to contribute towards building an Arabic automated sentiment analysis system. We aimed to strengthen some essential parts of the system to enhance the overall quality of the final outcome. The first part of this system to be strengthened is the sentiment lexicon which acts as knowledge base for the positive and negative tokens. The second part is the processing itself, and the the third part is the dimension reduction unit or stemmer. There are three main contributions in this thesis:

- The first contribution is introducing domain-oriented two manually constructed and annotated corpora and one sentiment lexicon. Those introduced items were tested by several statistical algorithms augmented with some lexical features [19].

- The second contribution is the development of automated system to construct domain-oriented lexicons using generative statistical models and equipped with overall-rated corpus [20].

- The third contribution is a dimension reduction algorithm introduced as an R package called arStemmer1. arStemmer1 is a powerful Arabic stemmer based on morphological and orthographic rules and supported with affixes lists constructed using neural network language model (NNLM).

This thesis consists of six chapters as follows:

**Chapter 1** is an introduction about sentiment analysis generally and sentiment analysis in Arabic language specifically and why we decided to work with Arabic instead of using cross
language methods and techniques. This chapter gives a brief about the unique characteristics of Arabic that make it unique not to be handled by the same algorithms used with English.

**Chapter 2** is a literature review focusing mainly on the Arabic resources that were built for sentiment analysis and the those stemmers which are for Arabic language specifically.

**Chapter 3** introduces our contribution towards Arabic sentiment analysis in the side of resources and benchmarking. It is about building corpora and lexicon and labeling them manually to be used as the base of stock market prediction application.

**Chapter 4** is an approach for domain-oriented sentiment lexicon construction using statistical models. The approach is the detection and extraction of sentiment tokens automatically using topic modeling algorithm and overall rated corpus.

**Chapter 5** discussed a new a algorithm for Arabic stemming as the main dimension reduction step. The solution is introduced as an R package called arStemmer1.

**Chapter 6** is the conclusion and some future work thoughts.
CHAPTER 2
LITERATURE REVIEW

2.1 Sentiment Resources

Sentiment analysis follows two main approaches: semantic approach (word-based), and statistical learning (machine learning), and for each of those methods, we need a type of resources. The most fundamental unit in the sentiment analysis system is its knowledgebase, lexicons for semantic approach, and labeled training data for statistical approach. Lexicon is a repository of words (tokens or text units) annotated with polarity (e.g. positive, negative, and neutral) or degree of polarity. The lexicon can also contain terms of emotions classes such as happiness and sadness [5]. The other kind of resources is annotated data set which is a group of documents (tweets, messages, reviews) labeled with polarity label, or emotions label for the purpose of direct usage or to train statistical algorithms. The sentiment resource labels can be absolute (positive or negative, zero or one) and can have different levels of granularity such as very positive, very negative, or associated with a numerical value representing its weight [21].

Sentiment lexicons can be created with many different ways, types and sizes. Lexicons can be created manually, automatically, or using semi-supervised approach [5]. SO-CAL and Sentiment Treebank lexicons are two examples of the manual ones, SentiWordNet lexicon was constructed automatically, and Macquaire semantic orientation lexicon was created using a
semi-supervised approach. We will go through some of the well-known English sentiments lexicons to give an idea about how they were constructed and then we will talk about Arabic lexicons.

2.1.1 English Lexicons

SentiWordNet

SentiWordNet associates three scores, positivity, negativity, and objectivity, to each synset of the English lexical database WordNet [22]. The score’s range is between 0 and 1 and the total of all the three scores is 1. The methodology used to build SentiWordNet is to train eight individual classifiers to classify synsets into one of the three classes: positive, negative, objective. Synsets were represented by vectors their gloss [22]. The training data was created by starting with seed synsets (positive and negative) and then propagate using the relations adopted by WordNet: antonym, similarity, derived from, pertains to, attribute, and see also [5].

SO-CAL

Semantic oriented calculator (SO-CAL) is a manual lexicon different from SentiWordNet as it doesn’t contain sense information [5]. This calculator was based on five dictionaries translated from English to Spanish. Those dictionaries contain 2,257 adjectives, 1,142 nouns, 903 verbs, 745 adverbs, and 177 intensifying words (such as very) [23]. The polarity of each word in SO-CAL is a number between -5 and 5.
Stanford Sentiment Treebank

This lexicon consists of about 215,154 phrases labeled manually by Amazon’s Mechanical Turk’s (crowdsourced) [5]. First, a corpus of movie reviews that has 10,662 sentences was parsed using Stanford Parser to result in 215,154 phrases. After that, each phrase was labeled by crowdsourcing approach [24]. This treebank was supported by a system called Recursive Neural Tensor Networks to detect single sentence sentiment.

WordNet-Affect

To build this lexicon, a group of synsets of WordNet lexicon were selected as an affective concepts representatives. There were two main steps, the first one is the creation of AFFECT, which is a manually initialized words set. The second step is the extension of seeds set based on the rules of WordNet. AFFECT contains 1,903 direct and indirect emotional terms, 539 of them are nouns, 517 are adjectives, 238 are verbs, and 15 of them are adverbs. Lexical information was added such as part of speech relation (POS), synonyms, antonyms, and the relation between English and Italian terms. The propagation from Affect through WordNet was based on simple rule: add synset that contain at least one word from Affect. The extension of WordNet-Affect can be done by utilizing the other relations provided by WordNet taking in consideration preserving the effective meaning [25, 26].

SenticNet

SenticNet is a single-, and multi-word concepts lexicon extracted from the large-scale
freely available knowledge base, ConceptNet [26]. ConceptNet is a huge data based about what is called common sense knowledge like: the sky is high! It is supported with natural language processing tool-kit for text reasoning purposes. 1.

ConceptNet has resourced its concepts from the a freely available natural language statements crowdsourced knowledge base called Open Mind Common Sense (OMCS) 2. In addition, ConceptNet has been enriched also by other resources such as WordNet and DBpedia (Wikipedia based structured information).

SenticNet is also merged with WordNet-Affect by converting both lexicons into matrices and then combine them. The concepts and polarity scores of SenticNet are encoded using Resource Description Framework (RDF) and represented by XML. SenticNet can be accessed using several way such as RDF/XML local file, API, and via python programming language package. SenticNet contains 6,122 single-word concepts (59% positive and 41% negative), 6,839 two-word concepts (55% positive and 45% negative), and 780 concepts of three words and more (48% positive and 52% negative) [26].

2.1.2 Arabic Lexicons

Arabic WordNet (AWN)

This Arabic lexicon was created using the same construction process of Princeton WordNet (PWN) and Euro WordNet (EWN) taking in consideration language-specific concepts and relations. Some concepts are added from other language and translated manually

1http://conceptnet.io/  
2https://launchpad.net/openmind
to the nearest Arabic synset. The database has four main principal entity types: item, word, form, and link [27]. A high level of generality is maintained on AWN, which means Arabic word has to be linked to the right English synset through finding all senses of the word, and from the English synset, the Arabic variants have to be found. AWN started by acquiring 1,000 nominal and 500 verbal synset from EWN and BalkaNet’s (WordNet for Balkan languages) CBCs (Common Base Concepts) [27].

AWN version 2.0.1 contains 11,269 synsets, 23,841 words, and about 161,705 links connected to the Suggested Upper Merged Ontology (SUMO) through concepts [28]. Some of the efforts done to enrich AWN are:

- Using PWN and AraMorph bilingual dictionary to add new synonyms to AWN. A mix between vocalized and non-vocalized terms were discovered and as a result, the coverage of Arabic words was reduced [29]. http://www.nongnu.org/aramorph/english/index.html

- Building Arabic dialect WN using WN for Modern Standard Arabic (MSA) and English-Iraqi dictionary [28].

- Using Wikipedia with some morpho-lexical patterns to enrich the semantic relations of AWN by addressing the problem of missing words and semantic relations between synsets [30, 31].

- A group of enrichments were done by [30] includes verbs and nouns, broken plurals, and entities and relations. For verbs enrichment, verb senses in AWN were extended using translation of English VerbNet [32] and Unified Verb Index (UVI)\(^3\) using rules of EWN.

\(^3\) http://verbs.colorado.edu/verb-index/index.php
In addition, AWN noun synsets hyponymy relations were refined using the technique of pattern discovery and Maximal Frequent Sequences (MFS) [30, 33]. To guarantee correct addition to AWN, the new enrichments were validated manually by lexicographers.

ArSenL

This Arabic sentiment lexicon was created by pairing the Arabic lemmas used in Standard Arabic Morphological Analyzer (SAMA) [34]) with three labels: positive, negative, and neutral. The main methods used to construct ArSenL are using the Arabic WordNet or using English glosses of a dictionary. The resources used to build ArSenL are WordNet, AWN, English SentiWordNet (ESWN), and SAMA. For the first method, Using The Arabic WordNet, the processes goes as follows: First, the scores were extracted by using the mapping between AWN and ESWN. Then, the correct lemma form were extracted by mapping AWN with SAMA. The mapping between AWN and ESWN were limited to verbs and nouns as there were no map files for adjectives. The mapping between SAMA and AWN has some issues such as the disagreement on lemma orthography in sometimes [35], and AWN has multi-word lemmas which is not the case in SAMA. The issue of multiple lemmas was solved simply by ignoring words with multiple lemmas. About 5,002 lemmas (7,326 synsets) in AWN were linked to 4,507 in SAMA. For the second method, English gloss-based, the process goes as follows: for each entry of SAMA lemmas, the English glosses of SAMA’s lemma entry are matched with the glosses of ESWN. The results found from method 1 and method 2 are combined to give the final result, ArSenL. The entries of ArSenL are distinguished by fields.
SIFAAT

Sifaat is a manually created Arabic lexicon that contains 3,325 adjectives with three labels: positive, negative, and neutral. Sifaat adjectives were taken from Penn Arabic Treebank (PATB) [36]. To expand Sifaat, English glosses were added to Sifaat entries from three large scale English lexicons: SentiWordNet, YouTube lexicon (YT), and General Inquirer (GI). GI contains 11,788 terms classified manually into some categories [37]. YT is a list of words (29,991) taken from a corpus of YouTube videos comments [38]. The expanded Sifaat has 229,452 entries associated with their English glosses.

SANA

SANA is a large scale multidialectal multi-lingual lexicon for Arabic sentiment analysis and subjectivity. It covers MSA, Egyptian dialect, and Levantine dialect. The four main genres of SANA are: tweets, chatting posts, newswire, and comments taken from YouTube. SANA was developed using both manual and semi-automatic methods. The semi-automatic method includes direct translation from English to Arabic and using the word association calculation, pointwise mutual information (PMI). The manual method is extracting words from two Arabic manually labeled polarity resources: Sifaat, and HUDA [39]. HUDA contains 4,905 entries mined out of a big corpus of chatting messages (11 million chat turns). The categories of HUDA are: 1,900 positive terms, 1,080 negative, and 1,925 neutral. For the
For the semi-automatic method, a collection of 971,659 noisy labeled tweets was processed by PMI with five different thresholds of word frequencies. With the threshold 25, the result was 6,572 positive tokens and 6,157 negative tokens. PMI also applied on the corpus of Egyptian chat (11 million chat turns) with the same thresholds used with the tweets corpus. The selected threshold was 5 (word frequency) and the result was about 29,000 positives and about 22,000 negatives.

THARWA

Tharwa a three-way lexicon: Dialectal Arabic (DA), Egyptian Arabic (EA), and MSA. In addition, Tharwa contains English glosses and linguistic information such as part of speech (POS), number, gender, rationality, and morphological root [40]. Tharwa contains about 73,000 entries taking in consideration the morphological, phonological, and lexical differences between MSA and EA. In addition to SAMA, several resources were used to build Tharwa such as:

- BADAWI, [41] which is paper-based dictionary of EA-English translations and definitions. Badawi contains 31,548 entries of single words attached to their POS.
- Egyptian Colloquial Arabic Lexicon (ECAL) [42]: ECAL has about 66,000 Egyptian Arabic undiacretized entries.

- Columbia Egyptian Colloquial Arabic Dictionary (CECAD): it is EA-MSA-English lexicon containing 1,752 entries taken from ECAL and manually enhanced with MSA and English matches [40].

- CALIMA lexicon: CALIMA is EA morphological analyzer based on the unique lemmas of ECAL (about 36,000) and augmented with further enhancements such as POS mapping and morphological segmentation [40].

Tharwa is constructed out of all the mentioned resources with some standardization steps to overcome the format differences between resources. One of those standardization steps is adopting the Conventional Orthography for Dialectal Arabic (CODA) that was proposed by [43]. Moreover, Tharwa construction process included providing the MSA and English correspondents of the resources Badawi, ECAL, and CALIMA lexicon.

2.2 Stemmers

Stemming is mapping a group of inflected or changed words to their original form or base. This base can be the root, or can be the stem, which might be any other form but not the root such as the shortest form of the whole set of words or a substring of them. For sentiment analysis for example, a document might have several morphological forms of
positive words, but this set of positive words can be reduced to one word only which is the stem.

Stemming can be done using several approaches such as root-based stemming, light stemming, and dictionary-based stemming [44]. Root-based stemmer is based on pattern matching to find the root of a word, an example of root-based stemmers is Khoja stemmer [45] which is one of the two most successful Arabic stemmers (more about Khoja stemmer follows). Light stemming means removing affixations such as suffixes and prefixes. Dictionary-based method is about searching a dictionary to find the root of a processed word (processing means removing specific affixes) [44].

According to [46], Arabic stemming has four main approaches as follows: the manual construction of dictionaries, light stemming, morphological analyzers, and statistical stemmers which use clustering methods to group related words in one class. Arabic stemmers, and may be other languages stemmers, can easily fall in one of two mistakes: either missing some members of the stem word class, or adding unrelated members to the class [46]. Some of the very popular Arabic stemmers are listed below:

2.2.1 Khoja Stemmer

Khoja stemmer is based on two main steps, preprocessing and matching. In the first step, the text goes through several stages of cleaning by removing diacritics, numbers, punctuation, stop words, the definite article “ة” the ”, the conjunction “و and”, and affixation (prefixes and suffixes). In the second step, the word list from step one is matched against
a list of patterns and roots, and the match is taken as the root. One additional step is to replace the occurrences of Hamza ء with ی, and the occurrences of the letters "و و" with the letter و [47]. There are some weaknesses about Khoja stemmer. The first weakness is the continuous need of maintaining the dictionary as the language change. Secondly, the replacement step with the letter mentioned earlier the wrong roots. Third, Khoja failed to remove all affixes [47]. In addition to that, Khoja doesn’t not handle proper nouns correctly such as countries names.

2.2.2 ISRI Stemmer

The Information Science Research Institute’s (ISRI) Arabic stemmer is a root-based Arabic stemmer sharing some features with Khoja stemmer and overcoming the shortcoming of Khoja as ISRI does not need root dictionary [47]. In addition to preparations done by Khoja such as normalizing some letters (e.g. hamza )and removing some affixes (e.g. the conjunction and و), ISRI is based on two sets, set of patterns P, and set of affixes S. ISRS returns a stem of length four when the word is of length four. For words of length five, a first try is done to extract a stem of length three or four for the words that match patterns of the group PR53 (word length 5 and stem length 3) and PR54 respectively. The same attempt is done for the words of length 6 with group PR63. If the word length is seven then stripping one suffix and prefix is tried first and if succeeded, the word is then processed as of length six. More details about the lists of prefixes P, suffixes S, and patterns and roots can be found in [47]. The diagram below summarizes the whole process of ISRI stemming.
Figure 2.1: Khoja Stemmer.
2.2.3 Berkeley MT-Based Arabic Stemmer (BMTAS)

This stemmer was created by researchers from the University of California at Berkeley, and we call it Berkeley MT-Based Arabic Stemmer (BMTAS). The idea of this stemmer is to translate the Arabic words are translated to English, and then grouping them into clusters according to the English stems [48]. After that, the shortest Arabic word of each cluster is chosen to be the stem (morpheme) of all the words in the cluster. A morphological analyzer is used for English to change plural nouns into singular and verbs into the infinitive form, and to convert adjectives into the positive form (e.g. bad, worse, worst, the positive form is the word “bad”). Dealing with this problem is much easier in English than in Arabic because the broken nouns in Arabic are very hard to stemmed, for example the plural form of the singular word رجل Man, is for men, which can have so many suffixes or prefixes making it hard to strip them and change the form while the same word in English is separate and can be handled easily. In other words, when the Arabic word: for men man is translated to English, it removes the Arabic prefix, and then the available English morphological analyzer can be used to map the word, man to its stem man. So with translingual resources between English and Arabic, and good morphological analyzer in English, an efficient Arabic stemmer can be built [48].

2.2.4 Berkeley Arabic Light Stemmer (BALS)

A light stemmer introduced by the same researchers mentioned before [48] from the University of California at Berkeley, and we call it Berkeley Arabic Light stemmer (BALS).
The main idea of BALS is to strip affixes as shown in Figure 2.2. The affixes to be removed are listed in six lists, three for prefixes and three for suffixes. The prefixes lists contain the frequency of the initial, the first two, and the first three characters of Arabic words in a collection of text. For the suffixes lists, they contain the final, the last two, and the last three characters. Out of those lists, a group of affixes were selected to be removed according to empirical evaluation, grammatical functions, affixes frequency in the text, and the English translations of the affixes. BALS stemmer has many affixes in common with the stemmer done by [46] and the stemmer created by [49]. The stemmer BALS removes prefixes non-recursively and removes suffixes recursively.

Interestingly the light stemmer BALS showed better performance than MBTAS with respect to information retrieval, in the same experiment executed by [48].

2.2.5 TREC-2002-Enhancement

Arabic light stemmer was introduced by [50] as an enhancement for the stemmer TREC-2002 built by [49]. The enhancements included a new addition to the affixes list created by TREC-2002. There were about seven additions to the suffixes lists. For prefixes, there were excluded and added affixes. The four characters prefix لمبا was added as an enhancement. The comparison between TREC-2002 and TREC-2002-Enhancement is shown clearly in Figure 2.3 where common affixes are in the bold in the middle. Two execution approaches were used: suffix-prefix approach and suffix-prefix-suffix. In the first approach, suffixes are removed recursively, and when it is done, prefixes are removed in a non-recursive
Figure 2.2: BAL Stemmer.
way. In the second approach (suffix-prefix-suffix), two suffixes and one prefix are removed in the order: suffix-prefix-suffix.

2.2.6 Light10

Light10 is a light stemmer that removes the conjunction و “and”, a set of prefixes (ال، ان، ات، ون، ين، يه، ية، ه، ت، ي) and a set of suffixes (ها، ان، ات، ون، ين، يه، ية، ه، ت) making sure to leave at least two characters [51]. Light10 is very effective for information retrieval as it outperformed popular stemmers such as Khoja [51]. The effectiveness assessment was done using standard TREC (Text Retrieval Conference) data. Different sets of prefixes and suffixes were tried with a name for each try, as Light1, Light2, Light3, Light8, and then Light10.
2.2.7 Al-Stem

alight stemmer that strip prefixes and suffixes shown in the lists below: Prefixes: 
\[
\text{وال، بال، بت، يت، وت، ست، نتسبم، لم، وم، حكم، فم، ال، لل، وي، لي، في، وا، فا، لا، با}
\]
Suffixes: 
\[
\text{ات، الوا، ون، ون، ان، تي، ته، تم، حكم، هم، هن، ها، يه، تح، دا، ين، يه، ل، ل، ي،}
\]
There was a threshold of probability to accept the prefix and suffix as a valid affix in Al-Stem construction process [52]. Those lists were manually examined and filtered in accordance with the affixes lists done by [46].

2.2.8 Sebawai Stemmer

Sebawai is a cross-platform light morphological analyzer for Arabic language that is based on a list of word-root pairs [13]. This list is used to extract lists of prefixes and suffixes, create stems templates, and to compute the probability of every single entry in those lists (suffixes, prefixes, and templates). The word-root pairs list was generated using a pre-existing morphological analyzer called ALPNET [53]. The probability is calculated by dividing the occurrences of a prefix or suffix by the total number of words. To extract the root from an input word, Sebawai generates tuples of the form prefix-suffix- template (PST) by gradually removing suffixes and prefixes from the input word. Those generated PSTs are then ranked as possible roots (some are excluded by the system). The system has some limitations such as handling words with weak letters (\(\text{ا و ي}\)) which is addressed by adding one
more term to the probability calculation. The added probability is the likelihood of a letter to be substituted or doubled (doubling the second letter of a two-letters word to get the root).
CHAPTER 3
SENTIMENT ANALYSIS BASED FUZZY DECISION PLATFORM FOR STOCK MARKET

3.1 Introduction

One of the main applications of sentiment analysis is the prediction of stock market direction. Stock market prices are being discussed every day on microblogs, discussion forums, and stock markets websites; and a huge amount of text is produced everyday. In some research mentioned by [54], news and sentiments can drive the market significantly, and human decisions are affected by emotions and moods. According to a study done by [55] about the Saudi stock market, investors decisions can be irrational and cannot be described by the normal financial theories.

The Saudi investors, as other investors in other markets, tend to seek advice before investing in the market. In this chapter, we applied sentiment analysis on a huge amount of documents (forum posts and tweets) to make investors able to predict the Saudi Stock Market (SSM) movement. As the decision of investment is critical we counted on manually constructed two corpora and one lexicon as the manual way is more reliable and trusted than the automatically constructed ones as stated by [56].

There are two main contributions for this chapter. First, we introduced the following manually constructed objects: Saudi Stock Market Lexicon (SSML), an annotated Twitter
corpus, and an annotated forum posts corpus. Second, we proposed a rule-based fuzzy decision approach based on the sentiment analysis of those three objects. We adopted two main analysis methods: corpus-based approach and semantical-lexical based approach. We used recall as the performance measure because we focus on one class only as we will see later in this chapter, the class is either positive or negative as the customer might be interested in buying on a market dunk, or want to buy high and sell higher!! In other words, using recall is answering the question: are we correctly classifying a good number of the relevant posts? This number of classified posts can be divided by the total number of posts to know how positive/negative is the market. The real time prediction of the market is beyond the scope of this study.

The rest of this chapter is organized as follows. Section 2 and section 3 talk about the related work and data collection respectively. Section 4 is about text preprocessing, section 5 gives details about SSML, and section 6 is the experiment and results. The decision making is discussed in section 7, and we conclude by section 8.

3.2 Related Work

A text-based decision support system was proposed by [57]. Using a collection of financial text documents, they extract all sequences of events from documents and infer any possible hidden relations between them. The system has four main parts: text processing unit, textual information generalization, event sequence extraction, and classifier-based inference engine. They used the decision tree classifier to make decisions. Online opinions about the stock market were utilized by [58] to predict the stock market prices volatility.
They found that sentiment analysis of stock market posts is less accurate than the statistical machine learning methods. They used manually labeled documents and sentiment analysis to label the new documents. After that, they used indexed posts with Support Vector Machine (SVM) classifier to build their model. Their model consists of two main parts: SVM and Generalized Autoregressive Conditional Heteroskedasticity (GARCH). GARCH is used to model the financial time series and SVM is used to estimate GARCH’s nonlinear function between the time varying and auto-regression. The system proposed by [54] is a sentiment-based real-time system to monitor the market movements. The authors used variations of classifiers to classify posts taken from an investment forum called HotCopper. The main advantage of HotCopper is the sentiment annotation associated with each post. Naïve Bayes (NB) was employed to classify investors’ sentiments, and classification was improved by using the Term Frequency Inverse Document Frequency (TF-IDF) transformation to rank terms according to their TF-IDF values. The authors used Bernoulli model of Naive Bayes to test their classifier. About 7,200 features were selected including positive and negative bigrams and trigrams. The classification F-score was about 77.50%. In the research done by [59], it was found that stock market textual information is more important than the trading volume in predicting the market volatility of some firms (not for short period of time e.g. day). They used NB algorithm to classify 1,559,621 of stock messages into three main classes: BUY, HOLD and SELL. Then, they measure the bullishness of the market in a specific period of time by aggregating those classified messages into indices. All HOLD messages were ignored because the noise in this group dominates the neutral related messages. Among the measures the authors used is the activity which was measured in thousands of messages and
the intensity which is the average number of words per message.

A study by [60] has investigated the correlation between the general mood of people and the market behavior. The authors achieved about 76% accuracy using Self Organizing Fuzzy Neural Networks (SOFNN). SOFNN has two preprocessed inputs: the row Dow Jones Industrial Average (DJIA) values and the sentiment analysis of the tweets. The sentiments were categorized into four main classes: calm, happy, alert, and kind. From sentiment analysis perspective, the problem with this study is the domain of the sentiments. The data was general tweets talking about many things including the stock market, and it is not taken from more specific and related board to the market such as investment forums. In other words, instead of general sentiment such as “happy,” more related features can be selected (like SELL, BUY, Up, Down, etc.) to give clear sentiment about the market direction.

Arabic sentiment analysis is covered in a lot of research summarized by [56]. Some research adopted corpus-based models and some of them adopted lexicon-based models, and each group has some reasons for their decision. In [56], they have built a tool for Arabic sentiment analysis and introduced a corpus and a lexicon to the interested researchers. They measured the accuracy on different sizes of lexicons, different types of corpora and with and without stemming. From a decision making point of view, the problem with their study is the diversity and impurity of the data. They translated 300 words from English to Arabic as their first step to build the lexicon which might not be in the same level of quality as taking terms from Arabic resources. In addition to that, they talked about poor performance because of gathering more than one Arabic dialect in the same experiment (this might affect the decision making process seriously).
The closest work to ours was done by [61]. Hamed et al. collected 1,943 tweets and classified them using machine learning algorithms: NB, SVM, and KNN. They tried to find any relation between the sentiments and the market direction, and they got 24% of relation between the downside of the market and the negative sentiments, and they got 36% of relation between the upside of the market and the positive sentiments. The main difference between this work and our work is, we provided a robust lexicon for SSM, we included both investment forum posts (over long period of time) and tweets with much bigger numbers, and we used more methods of analysis and more performance measures.

3.3 Data Collection: Corpora and Lexicon

Since the decision of investing in the stock market is risky somehow, we decided to build our lexicon manually to assure more robustness, accuracy, and reliability. The author of this thesis, had a portfolio in the Saudi market for at least 5 years and he has enough experience to gather the key words in this matter. We constructed and labeled two corpora and one lexicon. The first corpus was taken from the very well known Saudi investment forum called Saudishares\(^1\), and the second one is taken from Twitter. The lexicon was built manually out of those corpora.

3.3.1 Saudishares Corpus

This forum corpus was collected from Saudishares forum, one of the biggest Saudi investment forums that has been going on for more than ten years now and has tens of

\(^1\)www.saudishares.net/vb/
thousands of comments and reviews about SSM. We collected 18,695 posts using the tool OutWit Hub\(^2\) with our customized macros and scrapers to navigate through all pages and pick only the bodies of the posts. We designed macros and scrapers to crawl through the forum and select the main posts for the last ten years (not the replies and side comments). We aimed to cover a long period of time to make sure we include all diversities of market terms and to cover all major events and crisis that happened in the Saudi market, such as the crash that happened in 2006, to have all levels of positivity and negativity features in our corpus. Most of the titles were included in the posts themselves, so we excluded all titles. After collecting the posts, we removed some missing values (rows that have almost nothing) to have 18,177 posts left. Then, those post were cleaned and annotated manually to positive, negative, and neutral categories. The number of posts of each class is as follows: 4,029 positive, 1,544 negative and 12,604 neutral.

3.3.2 Twitter Corpus

Twitter corpus consists of 8,940 tweets including missing (no text, just few symbols) values; after removal of missing values the number was 8853 tweets. The most popular domain-related Twitter accounts were selected to be on the focus of our corpus construction process. We have chosen about 30 Twitter accounts according to some indicators such as the number of tweets, number of followers, and the number of likes. These accounts in general give advices to the investors, like when to buy and when to sell, good and bad news, and so forth. We consider only some of those popular accounts who give textual

\(^2\)https://www.outwit.com/products/hub/
information, because some of them are just referring the followers to other websites that have charts explaining the movement of the market. We extracted tweets from Twitter using Twitter Application Program Interface (API) and the python library tweepy\(^3\). We employed the python tool developed by Yanofsky and available on GitHub\(^4\). As we did in the Saudishares corpus, we labeled Twitter corpus (8,853 tweets) manually taking in consideration the importance of the right decision in the stock market. To minimize the risk of false positive labeling (when used later in automated labeling), we label a tweet (and post) as positive if it is clearly saying something positive about the market such as a recommendation to buy, an encouraging announcement, or at least assuring the normal investment environment with no high risk. The same process was used for the negative documents; we make sure it is clearly negatively opinionated. In this corpus, there are 2,224 positive, 612 negative and 6,017 neutral documents.

3.4 Text Preprocessing and Features Reduction

In the preprocessing stage, we cleaned and processed both Twitter and Saudishares corpora to be annotated and used to train the models. The preprocessing stage went through the following steps: removing unwanted parts such as urls, removing stop words, removing non-Arabic characters, and tokenization.

We decided not to do stemming for some reasons. First, we work with Arabic informal text (Saudi dialect) which has to be processed in a special customized way as it has so many slangs, special abbreviations, and semi-permanent spelling mistakes\(\text{even native}\)

\(^3\)https://github.com/tweepy/tweepy
\(^4\)https://gist.github.com/yanofsky/5436496
speakers fall in them frequently). Some of those spelling mistakes are tricky, such as the word "Dilaal" which means shades; it can be mistakenly written as "Dalaal" which means misguidance. Second, we searched the literature and found that the Arabic stemmers are still suffering from serious weaknesses as mentioned by Larkey2006.

3.5 Saudi Stock Market Lexicon (SSML)

SSML is a domain-dependent sentiments lexicon constructed specifically for the Saudi stock market investors, or any other parties, to make decisions. SSML contains terms and their sentiment polarities (positive and negative) with two levels of weight, A and B, where level A is stronger than level B in both positive and negative polarities. Our lexicon is constructed manually by analyzing both corpora of Twitter and Saudishares forum. The lexicon is divided into two main parts: positive and negative. Each one of those parts is split into two sub-parts A and B. Part A contains the most unique words that can be used alone to give a very clear indication about the polarity of the document. As an example of these words is a word like "congratulation," "green," "excellent," or the word "ممتاز" the occurrence of such words in a sentence can give enough sign of positivity. It is not very common (as far as we know) to say "very green" or "not green." Part B is less intensive than part A as it might be affected heavily by other features such as negation or intensification (e.g. very, quite... etc.).

Negative lexicon part A has clear identifying tokens that work very well to separate the negative documents from others (as we will see shortly). This is because negative terms in the stock market are mostly mentioned in pure negative context, while some positive
words can be mentioned in both positive and neutral context. For example, the sentence: "good morning, صباح الخير," has no contribution towards the polarity of the tweet, but can be misclassified as positive because of the word "good," which is in the positive lexicon. Consequently, we supported the positive classification by other features such as the stock name, numerical values, and Arabic negation words list.

Our lexicon is reviewed and revised by two experts in stock market to make sure we included most of the key terms. Although our lexicon did a great job in the process of decision making in the Saudi stock market investment, we might do more enhancements in the future such as POS tagging and using some of the words as seeds to expand SSLM to include more Arabic sentiment words. That needs more manual work as there are some positive words in the stock market domain but neutral in other domains such as "green" or "entrance." The word “green” is positive sentiment associated with the up direction of the market (means the stock is increasing) and the word “entrance” gives an impression that the market is safe to invest in. The size of each section of our lexicon is shown in Table 3.1.

As we mentioned earlier, we work with un-stemmed tokens in this study. We preserved all forms of the word (including spelling mistakes that are somehow frequent).
specifically the key words as they have been used repeatedly in the stock market discussion forums. Although the difference between the number of stemmed and un-stemmed terms is big especially in group A, at this stage of our research, we care about the comprehension, and accurate decision more than the size of the text being processed. A light stemmer might kill a very useful key word such as ”happiness.” The word happiness in Saudi dialect is ”/hehfinal/seeninitial/aleffinal/nooninitial/wawisolated,” both of the words ”/hehfinal/seeninitial/aleffinal/nooninitial/wawisolated, happiness,” and ”الناس, people,” are mapped to the same stem: ”/seenisolated/aleffinal/noonmedial/laminitial/alefisolated, الناس, people.” This means some neutral words might be taken as key words or key words might be treated as normal words depending on what is in the lexicon.

3.6 Experiments

Our experiments went through two main stages as in the following sections. We tried a variety of algorithms, data sizes, and data types to select the best and make it the cornerstone of our decision making process.

3.6.1 Corpus-Based Approach (CBA)

Support vector machine (SVM) was employed to classify documents in both tweets corpus and Saudishares corpus as it is recommended by some resources, such as [62], as the best way to classify text. We provided SVM with manually annotated corpora (about 80% of the data for training) and we got the results shown on Table 3.2. We tried several combinations of the following items of data: posts or tweets, two or three classes, and balanced or unbalanced data. In our study, we care about recall, precision, and f-score more than accuracy. As we can see in Table 3.2, the best F-score we got is 63% and the best recall
is 64%. We used the R package RTextTools, developed by [63], that has the function SVM implemented in it. The SVM arguments used are: Kernel= radial, Gamma=1, and Cost=1. More details about how SVM works can be found in [64]. In the following section, we will use the same data but with our designed rules and compare it with CBA. We will focus on how to get the best results for investors to make the right and safe decision.

3.6.2 Semantical and Lexical Based Approach (SLBA)

In this section we used two approaches, statistical approach (machine learning) and rule-based approach with both semantical and some lexical features as follows.

Statistical Approach

We used SSML assisted with some lexical features (e.g. digit characters) to classify documents (tweets/posts). We employed several supervised machine learning algorithms mentioned by [65] and trained them (with 80% of the data) to be able to classify our documents into one of the three categorical responses: positive, negative, and neutral. We
used Linear Discriminant Analysis (LDA), Quadratic Discriminant Analysis (QDA), Neural Network (NNet), K-Nearest Neighbors (KNN), Penalized Discriminant Analysis (PDA), Decision Tree C5.0 (C5.0), and Random Forests (RF).

Before we apply those machine learning algorithms, we first generated several data frames based on our lexicon SSML and the lexical features. We extracted several features from the documents and we selected the top eight qualifying predictors (variables) as listed below. Those predictors were selected according to the Best Subset Selection (BSS) method explained by [64], which fit a least squares regression for each possible set of p predictors. The top selected attributes are as follows:

- Document length: number of tokens per document.

- Noise similarity: how similar is the document to the noise set, where noise set is a set that contains the useless words.

- Positivity A: the intersection between the document and the positive lexicon A.

- Positivity B: the intersection between the document and the positive lexicon B.

- Negativity A: the intersection between the document and the negative lexicon A.

- Negativity B: the intersection between the document and the negativity lexicon B.

- Digits occurrence: categorical value (0 or 1) to indicate whether the document contains numbers or not. This is useful because the nature of tweets and posts in the Saudi stock market might include a very short advice consisting of a number and very few
Table 3.3: Accuracy of Statistical Learning Algorithms for Both Tweets and Forum Posts With Different Sizes.

<table>
<thead>
<tr>
<th>Data Sets</th>
<th>Algorithms Accuracy</th>
<th>LDA</th>
<th>QDA</th>
<th>PDA</th>
<th>RF</th>
<th>C5.0</th>
<th>KNN</th>
<th>NNET</th>
</tr>
</thead>
<tbody>
<tr>
<td>Raw Posts</td>
<td>3 Classes</td>
<td>Balanced</td>
<td>66%</td>
<td>63%</td>
<td>67%</td>
<td>63%</td>
<td>67%</td>
<td>61%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>74%</td>
<td>72%</td>
<td>74%</td>
<td>77%</td>
<td>76%</td>
<td>63%</td>
</tr>
<tr>
<td></td>
<td>2 Classes</td>
<td>Balanced</td>
<td>81%</td>
<td>79%</td>
<td>81%</td>
<td>80%</td>
<td>80%</td>
<td>77%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>84%</td>
<td>84%</td>
<td>84%</td>
<td>85%</td>
<td>85%</td>
<td>83%</td>
</tr>
<tr>
<td>Cleansed Posts</td>
<td>3 Classes</td>
<td>Balanced</td>
<td>67%</td>
<td>64%</td>
<td>67%</td>
<td>66%</td>
<td>67%</td>
<td>61%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>75%</td>
<td>73%</td>
<td>75%</td>
<td>76%</td>
<td>76%</td>
<td>74%</td>
</tr>
<tr>
<td></td>
<td>2 Classes</td>
<td>Balanced</td>
<td>81%</td>
<td>79%</td>
<td>81%</td>
<td>82%</td>
<td>80%</td>
<td>79%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>84%</td>
<td>84%</td>
<td>84%</td>
<td>85%</td>
<td>85%</td>
<td>84%</td>
</tr>
<tr>
<td>Raw Tweets</td>
<td>3 Classes</td>
<td>Balanced</td>
<td>65%</td>
<td>65%</td>
<td>65%</td>
<td>67%</td>
<td>65%</td>
<td>57%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>73%</td>
<td>70%</td>
<td>73%</td>
<td>74%</td>
<td>75%</td>
<td>71%</td>
</tr>
<tr>
<td></td>
<td>2 Classes</td>
<td>Balanced</td>
<td>80%</td>
<td>80%</td>
<td>81%</td>
<td>77%</td>
<td>80%</td>
<td>75%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>86%</td>
<td>85%</td>
<td>87%</td>
<td>86%</td>
<td>86%</td>
<td>86%</td>
</tr>
<tr>
<td>Cleansed Tweets</td>
<td>3 Classes</td>
<td>Balanced</td>
<td>67%</td>
<td>68%</td>
<td>67%</td>
<td>71%</td>
<td>66%</td>
<td>62%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>73%</td>
<td>72%</td>
<td>73%</td>
<td>76%</td>
<td>76%</td>
<td>73%</td>
</tr>
<tr>
<td></td>
<td>2 Classes</td>
<td>Balanced</td>
<td>78%</td>
<td>79%</td>
<td>78%</td>
<td>81%</td>
<td>79%</td>
<td>76%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Unbalanced</td>
<td>87%</td>
<td>86%</td>
<td>87%</td>
<td>84%</td>
<td>85%</td>
<td>83%</td>
</tr>
</tbody>
</table>

words. The words represent the name of the target stock and the number represents the next price target that the stock expected to reach.

- Stocks occurrence: a number to quantify how many stocks targeted by the document.

This is a good indication of positive sentiment as it is a common way in the Saudi share market forums to give advices to invest in some stocks. For example if the tweet is: “SARCO 31.90,” that means, SARCO is expected to go up to reach the next price level at 31.90 very soon.

We repeated the experiment by applying the machine learning algorithms mentioned in Table 3.3 on the defined predictors many times with a different data set in each time. The data sets are formed according to either the data is clean (processed) or not, balanced or not and if it contains two or three classes (two classes means positive and negative). The results
are listed in Table 3.3. It is noticeable that the accuracy difference between two and three classes datasets is between 10% and 20%, two classes’ data is more accurate. Although we removed all kinds of unwanted characters and symbols, that did not enhance the accuracy that much (less than 2%). The accuracy of unbalanced data (one class is a majority) is higher than the accuracy of balanced data. As we can see from LDA accuracy, when we deal with two classes only, the accuracy is higher than the case of three classes in both posts and tweets. The unbalanced data in this case, always give higher accuracy than the balanced data, which might be misleading sometimes as we mentioned before. For this reason, we count on other measures when it comes to decision making later in this chapter.

Rule-Based Approach

Rule-based classification means using IF-THEN conditions to classify data. This process goes through three main stages as stated by [66], rule creation stage, rule ranking measure stage, and classification stage, which will be shown in our algorithms shortly. Our rules are divided into two categories; one is to deal with the two classes’ data set and the other one is to deal with the three classes’ data set.

Three Classes Rules: In the three classes’ data set, we included the three labels: positive, negative, and neutral. The problem here is when a document is not classified as positive or negative, it falls in the big (default) class which is neutral. The size of neutral class is much bigger than the other two classes (69% in posts and 68% in tweets). This means big number of correctly classified documents as neutral and a high accuracy as a result. Because of that, the accuracy might not be the best metric for decision making
Table 3.4: Rule-Based Classification Performance for Data Set of Three Balanced Classes

<table>
<thead>
<tr>
<th></th>
<th>Accuracy</th>
<th>Recall</th>
<th>Precision</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Posts</td>
<td>Positive</td>
<td>70%</td>
<td>81%</td>
<td>66%</td>
</tr>
<tr>
<td></td>
<td>negative</td>
<td>74%</td>
<td>81%</td>
<td>71%</td>
</tr>
<tr>
<td>Tweets</td>
<td>Positive</td>
<td>73%</td>
<td>74%</td>
<td>73%</td>
</tr>
<tr>
<td></td>
<td>negative</td>
<td>76%</td>
<td>69%</td>
<td>81%</td>
</tr>
</tbody>
</table>

Recall = \( \frac{TP}{TP + FN} \) \hspace{1cm} (3.1)

Precision = \( \frac{TP}{TP + FP} \) \hspace{1cm} (3.2)

Accuracy = \( \frac{TP + TN}{TP + TN + FP + FN} \) \hspace{1cm} (3.3)

\( F_{score} = \frac{2 \times \text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \) \hspace{1cm} (3.4)

The trader might be interested more in market positivity or negativity, depending on his eagerness to buy or sell. Because of the indistinct boundaries between those three classes and because it is more useful towards decision making to focus on one class only, we adopted a method called One-vs-All (OVA) classification, that was explained by [68] to classify documents into two categories: target and otherwise. The target category is either
positive or negative depending on what we are trying to measure. If the investor is interested
in negativity more than positivity to be able to buy shares with low prices, then in this case
the two categories are: negative and otherwise. When the target class was the positive, we
sampled 8,058 documents out of the Saudishares corpus having 50% of them positive and
50% otherwise, and we sampled 4,448 documents of the tweets corpus divided into two equal
halves positive and otherwise. We used 10-fold cross validation to get the best results. The
same process is repeated as well for the negative class as the target class.

As shown in algorithm 1 in the Appendix, our algorithm consists of three main parts:
feature extraction part (steps 3-13), scoring part (steps 14-16), and documents labeling part
(steps 17-20). The input is a cleansed corpus of documents (tweets or posts), and the
output is annotated corpus. Using the concept of bag-of-words, we utilized our lexicon to
compute the values such as posA, posB, negA, negB and so forth. The variable posA for
example means the occurrences of words from the positive lexicon part A in the document.
The other abbreviations are: positive lexicon A (PLA), positive lexicon B (PLB), negative
lexicon A (NLA), negative lexicon B (NLB), companies’ names list (CL), negation words list
(NegationL).

The values C1, C2, th1 and th2 are integer values chosen carefully to maximize the
separability between target and otherwise classes. As an example of the best combination
of those values is the set of \{3, 6, 4, 1\} for \{C1, C2, th1, th2\} respectively. From the
contingency matrix (confusion matrix), we noticed that the negative class is more separable
than the positive class, as indicated by the precision. When the target is the negative class,
we had higher precision which means that the negative lexicon is more robust than the
positive one. In such noisy data, we preferred to learn only one target class (recognition-based or one-class approach), as stated by [69], to resolve the problem of imbalanced data and to help to make the right market decision. Moreover, our choice to deal with the data as it (three classes, imbalanced, noisy) is to be as close as possible to the real world data in the Saudi stock market.

**Two Classes Rules:** In algorithm 2 shown in the Appendix, we handled only positive and negative classes. Rules were built to sharpen the boundaries between positive class and negative class as much as possible. We developed rules for both positive and negative classes but we showed the positive class in algorithm 2 as an example. We did the experiment several times with balanced (same number of positive and negative documents) and unbalanced data. For the Saudishares balanced corpus, we had 3,088 posts, 50% positive and 50% negative, and for the unbalanced Saudishares data, the number of positive posts was 4,029 and the number of negative posts was 1,544 (5,573 total). For the tweets, we had 1,224 tweets for the balanced data set, and 2,836 for the unbalanced data set (2,224 positive and 612 negative).

Although the rule is heavily driven by the lexicon SSML, the other features also give a good support and enhancement towards the decision quality; for example, the feature repL shown in algorithm 2, represents the occurrence of repeated letters (such as the word gooood) in the document. The percentage of documents that contain repeated letters in the tweets corpus (positive and negative class only) for example is about 13% of the documents number. Another feature is the occurrence of digits in the document. It is mostly a clear sign of an advice being given to the investors.

The balanced data gives less accuracy than the unbalanced, which is expected as the
imbalance ratio (defined by [69] as minority class/majority class) is high. The positive class is majority in both posts and tweets; and as a result, the imbalance ratios are about 38% and 27% in posts and tweets respectively. The recall scores are very high for both tweets and posts (between 96% and 98%), and the minimum accuracy was 68% of balanced tweets data set (see Table 3.5).

Table 3.5: Rule-Based Classification Performance for Data Set of Two Classes: Positive and Negative.

<table>
<thead>
<tr>
<th></th>
<th>Posts</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Balanced</td>
<td>70%</td>
<td>96.50%</td>
<td>63%</td>
</tr>
<tr>
<td></td>
<td>Unbalanced</td>
<td>82%</td>
<td>96.50%</td>
<td>81.80%</td>
</tr>
<tr>
<td></td>
<td>Tweets</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Balanced</td>
<td>68%</td>
<td>96.70%</td>
<td>61%</td>
</tr>
<tr>
<td></td>
<td>Unbalanced</td>
<td>85%</td>
<td>97.70%</td>
<td>85%</td>
</tr>
</tbody>
</table>

3.7 Rule-based Fuzzy Decision

With the method rule-based approach, we got the best results with respect to the recall metric. According to that, we adopted rule-based approach as our way to make decisions. Since we are dealing with some uncertainty of the market risk level, the theory of fuzzy sets that was introduced by [70], can be employed as the decision making mechanism in this chapter. The fuzzy logic is able to handle the partial truth, which means that the truth is not only either zero or one, it can be true up to a limit. In other words, the membership of an element x to a set U can be partial instead of making strict decision either it is a member or not. For example, instead of saying today is cloudy, we can make fuzzy decision by saying today is 70% cloudy and 30% sunny.
Now let S be the set of safe decisions. We call this set fuzzy, because the degree of membership to this set is not binary (yes or no), it has some degree of membership as explained in the by [71]. So the items of this set (decisions) can have membership degree in the interval $[0, 1]$, where zero is not very safe at all, 1 is very safe, and some decisions fall in between. The membership function of the set S is denoted by $\mu_S$. It can be written as $\mu_S : X \rightarrow [0,1]$, Where X is the universal of the fuzzy set. As in the decision diagram shown in Figure 3.1, the corpus has some degree of positivity and some degree of negativity between 0 and 1, depending on the percentage of positive and negative documents in the corpus. So to make a decision about corpus D polarity, we use equation 3.5, where T refers to the target class, and O refers to the otherwise class.

$$\mu_S(D) = \max \left[ \mu_T(D) - \mu_O(D), 0 \right]$$

(3.5)
As the difference between the two sets (target and otherwise) gets higher, the level of decision safety gets higher as well. Again, the target of the investor might be the positive class or the negative class, depending on their investment plan. Some investors look at the market downside as an opportunity to buy.
3.8 Conclusion

In this chapter we manually built our own lexicon and labeled corpora specifically for the domain of Saudi stock market. The lexicon and the two corpora were analyzed using two methods: corpus-based approach and semantical and lexical based approach. We have selected the rule-based approach as the best way to classify documents. The classified documents were used as an input to a fuzzy decision mechanism for the Saudi stock market. We counted more on recall than any other metric as it is more realistic towards our goal. In the future work, we will apply our strategy over a specific period of time and compare our results with the real movement of the Saudi market. Some enhancements might include more noise filtering, more features extraction, and lexicon expansion.
CHAPTER 4
ARABIC DOMAIN-ORIENTED SENTIMENT LEXICON CONSTRUCTION USING LDA

4.1 Introduction

Sentiment lexicon is crucial for sentiment analysis. The lexicon can be general sentiment lexicon or domain-oriented sentiment lexicon. The general one contains words with unchanged polarities in almost all domains such as the words “wonderful” and “ugly”. The domain-oriented lexicon contains words that have very strong polarity in their domain while they might be neutral in other domains such as the positive word “green” in the domain of stock market.

Sentiments lexicons can be constructed using several ways. Some sources divided the process of sentiment lexicon construction into two main methods: thesaurus-based and corpus-based [72]. Thesaurus-based approach is about propagating the polarity of a seed word to the related words in the thesaurus, and corpus-based approach utilizes the co-occurrence of words in a corpus or document as an indication of having the same polarity.

A variety of resources can be used to build sentiment lexicons. Some of those resources are [73]:

- Thesauruses: using seed words (search terms), the thesaurus can be searched for syn-
onyms and antonyms. One of the best examples of available thesauruses is WordNet database, which can be used for both Arabic and English.

- Linguistics rules and heuristics: the language rules and grammars can be used to build a sentiment lexicon out of a collection of text. Rules are like part of speech, sentence structure, capital letters, conjunctions, and so forth.

It is easy to find labeled documents and reviews online about a product or a service. The overall label (positive or negative) of the document, such as stars or thumps, can be enough to classify all tokens in the document as positive or negative depending on document level polarity. To select only the key words as much as possible and not include all words, some further steps can be done such as removing stop and frequent words, and using some linguistics rules to extract the most powerful terms out of the document/corpus. External resources can be used as well, for example, if we use AND rule, we can utilize a search engine to find the most occurring words with a seed word. For instance, the word ”good جيد”, can be used as a seed word to find a big number of words that frequently co-occur with it. That is possible with some search tricks such as searching a website (or corpus) with the quoted phrase: “good and”. But the problem with this approach is, it still need some manual work to pick the right seed words and to inspect the results of the search engine result as it can be sometimes unexpected. As an example of that in Arabic is the key adjective: ”beautiful جميل”. This word is a very common person name in Arab countries which might affect the domain-specific lexicon with words that have no contribution towards polarity.
Domain-oriented lexicons are more efficient than the general purpose lexicons as the polarity is heavily driven by domains, and it is impossible to include all domains sentiments in one lexicon [73]. The strength of the domain specific lexicon is its ability to classify the domain related documents with high accuracy. The challenge of domain-oriented lexicons construction is the need to automatically dive in the domain itself to get its unique key terms (manual option is possible but hard). Because of that, we focused in this study on domain-oriented lexicon construction instead of general lexicons. We made a very essential assumption in this study which is the assumption of having human judged corpus of documents. In other words, we build our lexicon out of a corpus of documents (comments, tweets, messages, reviews,...etc) labeled by humans (direct annotation or through crowd-sourcing).

Most of the research in this area focused on English while Arabic is still less studied. In this chapter, we proposed a simple method to construct Arabic domain-oriented sentiment lexicon by adopting a topic modeling approach called Latent Dirichlet Allocation (LDA). We will compare this lexicon with a manually constructed one in the task of documents classification. Domain-oriented, domain specific, and domain driven lexicon will be used interchangeably through this chapter. As per our knowledge, this is the first study to adopt a topic modeling technique for sentiment lexicon construction in Arabic language. The rest of the chapter is organized as follows: Section2 reviews related work. Section3 introduces LDA. Section4 describes our model. Section5 is about the experiment, and we concluded by the final comments and future work.
4.2 Related Work

Lots of studies and research have been conducted about sentiment lexicon construction using both manual and automatic methodologies for Arabic and English languages. But Arabic language is still experiencing a serious shortage in this matter when compared to English. One of the methods used to build Arabic sentiment lexicon is using thesaurus with semi-supervised learning [74]. Arabic WordNet were used to build a lexicon containing 7.2K words with sentiment score, part of speech (POS) tag, diacritic marks for each term, and some terms definitions as well. Seed words were translated from English to Arabic (4 positive and 4 negative) and used with semi-supervised learning and Arabic WordNet relations to propagate the polarity of those seeds throughout Arabic WordNet database. Task-based was used to evaluate the final lexicon using two manually annotated Arabic corpora. The machine algorithms used were Support Vector Machine (SVM) and Naïve Bayes (NB).

Automatic Arabic lexicon construction methods were introduced by [75]. They adopted two automatic methods based on a labeled balanced corpus of comments: the first one is a direct translation from English to Arabic, and the second one is using term frequency to produce two mutually exclusive terms lists, positive and negative. In the second method, they generated 2,075 positive words and 6,543 negative words. The evaluation method of their approach was not clear enough if it was documents classification or direct comparison between lexicons.

Al-Moslmi et al. [76] introduced several resources for Arabic sentiment analysis. The first one is a corpus called Multi-domain Arabic Sentiment Corpus (MASC) which contains
manually annotated customers reviews from multiple domains and can be used to evaluate sentiment analysis methods in both domain-specific and general purposes. The second resource is a sentiment analysis lexicon that contains 3,880 positive and negative synsets associated with labels of POS, polarity, dialects glosses, and other the other forms of the lexicon’s entries. Several classification methods and feature sets were used to test the quality of the lexicon. According to Macro-F score test, the best performing classifier was SVM.

Very useful rules and guidelines for Arabic language lexicon construction were mentioned by [77]. Although this study was not about sentiment lexicon, it gave the framework for extracting sentiment lexicon from a corpus. The lexicon can be general purpose or domain oriented according to the corpus in hands. They introduced more than 50 rules for identifying both nouns and verbs. Those rules can be very helpful for several purposes such as part of speech tagging (POS), stemming, and sentiments analysis (when there is a labeled corpus ready).

To construct a domain-specific lexicon, one of the methods used is to utilize the domain corpus to calculate the semantic association between sentiments words, extract some contextual and morphological rules between sentiments words, and then propagate those rules through the whole corpus [78]. As a first step, thesaurus was used to detect the text parts containing sentiment words. After that, dependency between text parts (sentences/chunks) were utilized to filter the collection more. The sentiment polarity then propagated from sentiment seeds to candidate sentiment words using point-wise mutual information PMI. The rules used by [78] were either contextual rules such as AND rule and negation rule, or morphological rule such as the suffixes “ful” and “less” in hopeful and
hopeless respectively.

A method called double propagation was proposed by [79] to construct domain sentiment lexicon. It is called double because they work on both sentimental seeds and features. First, sentiment words and features are extracted from seeds lexicon. Then, those words and features are used to find more sentiment words and features. The process goes on until there are no more new words. The extraction step utilizes the relations between words, between features, and between words and features. The main idea of polarity propagation is to use contextual rules. Two words occurring in the same level (document, corpus, feature,......) can have the same polarity.

Latent Dirichlet Allocation (LDA) was employed by [80] to build domain dependent lexicon. They used some seed words to support LDA to get good results. To test their lexicon, they compared it to general purpose lexicons such as MPQA and GI. The problem with this study is the lack of clarity in some aspects. For example, the selection of the number of topics for LDA was not clarified clearly. Moreover, the abstract talks about domain independent lexicon construction and the experiment talked about domain-specific lexicon. Our approach is different from this approach as we built our lexicon without using seed words and we got higher accuracy.

LDA was also used by [81] to introduce a model called sentiment-aware LDA model (sLDA). The model sLDA utilized domain-independent seed words to enhance the model’s topic generation process, and assumed the number of topics to be 2 sentiment topics and K non-sentiment topics. Our approach is different from this approach as we didn’t use seed words, we fixed the number of topics, we took the sparsity problem(weakness of LDA) in
consideration, and we got better results for documents classification. In addition to those differences, we work with Arabic language which is different from English in many aspects.

We assumed that each word in the corpus has a polarity. This assumption is supported by the fact that our documents are aggregated and focused on one polarity either positive or negative which simplify the tricky step of LDA, topic number selection, and save the effort of any further manual inspection.

4.3 LDA

One of the most powerful topic models today is called Latent Dirichlet Allocation (LDA) [82]. Topic models are statistical models used to infer hidden structures from a collection of data using the observed features of that collection (words). For example if we apply a topic model to a collection (corpus) of text documents, we would be able to know what are the topics covered in that collection without taking the hard option of reading all those documents! LDA has been used in many applications related to discrete data including text collections [82]. LDA was used in so many fields and applications that has the textual flavor; for example, LDA was employed by [83] as a method to recommend tags (labels) for the users of tagging systems (e.g. Flicker), which enhances both the organization and search of web content. Moreover, LDA is also used in software maintenance where a corpus of code lines pushed in LDA (with information retrieval hint) to localize the bugs in the code (classification problem) [84].

LDA is not only associating words to each other as done by the other methods (such as mutual information and chi-square) mentioned by [85], but also associates them to topics.
When LDA process documents, then the observed features are words. To understand LDA, let’s assume a blank page is to be filled with words using LDA according to the following steps [86]:

- Randomly choose a distribution over topics, in other words we have the recipe of topic proportions for the document. For example, 50% of the page is about sports and 50% is about politics.

- For each word in the document do:
  
  - Randomly choose a topic from step 1. (the selected topic is chosen from the per-document distribution over topics)
  
  - Randomly choose a word from the corresponding distribution over the vocabulary. (Each word in each document is drawn from one of the topics).

This gives us the main characteristic of LDA: all documents in the collection share the same topics but with different proportions in each document. Words are the only observed features of LDA model while the hidden features are topics (per-document topic distributions, and the per-document per-word topic assignments). The input of LDA is a collection of documents (pages, tweets, reviews, post, etc) and the output is the topics covered in that collection and key words of each topic. The number of topics has to be specified manually and the output looks like the shape in Figure 4.2.

Topic names are not given by LDA but the user can name each topic according to
the set of words produced for each topic. The key idea behind LDA is that, topic words are always come together in the same space, as mentioned by [87]: “You shall know a word by the company it keeps.”

To be more formal let’s have a look on the main parameters of LDA and how they contribute towards the process of topic discovery. The parameters used are:

- $M$: documents.
- $N$: words.
- $\theta$: the topic distribution for document $i$.
- $\alpha$: controls topic distribution per document.
- $z$: the topic for the $j$th word in document $i$.
- $\beta$: controls word distribution per topic.

If $\alpha$ increases that means documents are more similar to each other [88]; and when $\beta$ increases, topics appear more similar to each other (may be it is one topic !). LDA is represented in three levels: corpus level ($\alpha, \beta$) which is assumed to be sampled once during the generation of the corpus, document level ($\theta$) which is sampled once per document, and word level ($Z, W$) which is sampled once for each word in each document. Choosing parameters is done by:

- Choose $N$ according to Poisson distribution.
- Choose $\theta \sim \text{Dir}(\alpha)$. 
For each of the N words:

- Choose a topic using Multinomial ($\theta$)

- Choose a word $w_n$ from $P(W_n|Z_n, \beta)$, a multinomial probability conditioned on the topic $Z_n$.

The joint distribution for all parameters is given by:

$$p(\theta, z, w|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n|\theta) p(w_n|z_n, \beta)$$  \hspace{1cm} (4.1)

The parameter $\theta$ is k-dimensional Dirichlet random variable, the dimension means simply the number of topics which can be one or more.

4.4 Model Description

We used LDA as shown in Figure 4.2 where the input is a corpus of positive and negative documents, and the output is two sets of key words for each category (positive and negative). Instead of topic inference, we utilized LDA to infer the polarities of each document and give two exclusive lists of the top key words of each polarity (category). So, we modified the graphical model of LDA [82] to have the model shown in Figure 4.1, and we call it polarity LDA (pLDA).

The main challenge of using LDA (and other topic modeling techniques such as PLSA) with short text is the sparsity of content [89]. To alleviate the problem of sparsity, we aggregated positive documents (posts) in one big document, and the same thing for negative
posts. This approach was adopted by [90] to aggregate tweets by account. Topic 1 and topic 2 in Figure 4.2 are simply the positive and negative key words ordered according to their importance in the context. We tried different number of terms from each class (positive and negative) starting from 100 up to 10,000 terms. For each number of terms, we measured the performance metrics as shown in Table 4.1.

We used two different data sets to test our model, posts and tweets. The data sets were created by [91]. More details about the data will be in the experiment section. In the
posts data set, we can see continuous increase in accuracy, precision, and recall as shown in Table 4.1. On the other hand, we haven't noticed that big difference in the tweets data set. This difference between posts and tweets data sets is because tweets are short compared to posts (data gathered when the tweet limit was 140 characters). One more reason is because the LDA lexicon was built out of posts corpus which makes posts the main domain and tweets considered to some extent to be a sub domain. In addition to that, posts were collected for a long period of time (around 10 years) which represents some transformation in the language used, while tweets are very recent. For both data sets, we got very good results. We compared LDA-based lexicon with two different lexicons as shown in Table 4.2. One is constructed manually, we call it manL, and the other is constructed based on the terms frequency on both positive and negative corpora, we call it freqL.

We constructed the freqL lexicon by selecting terms according to the ratio between
their frequency in positive documents and negative documents. So for a word $w$, if it occurred $fp$ times in the positive side, and $fn$ in the negative side, then the ratio is calculated as $fn/fp$. We have chosen the threshold to be 1 as shown below:

$$
Label = \begin{cases} 
\text{positive, } & \text{Ratio} \leq 1 \\
\text{negative, } & \text{Ratio} > 1 
\end{cases}
$$

The lexicon freqL contains 5162 tokens. After manual inspection of the words list, the threshold selected was 1. This threshold splitted the lexicon to positive and negative parts. The positive part has 4,138 terms and the negative has 1,026 terms. The manually constructed lexicon is the Saudi stock market lexicon (SSML) that was created by [91]. It contains about 1,886 positive words and 1,975 negative words. The comparison shown in Table 4.2, is with LDA with 2000 terms taken from each polarity. Although the manL lexicon outperformed LDA-lexicon in tweets dataset, the manual work is still not easy to conduct compared to LDA. LDA-lexicon outperformed the manual work in the posts data set which is supported by the length of posts compared to tweets. Although freqL showed excellent performance compared to the other two, it needs manual inspection to pick the right threshold. From the tweets dataset analysis shown in Table 4.2, we can see that the manual lexicon outperformed LDA lexicon.

4.5 Documents Preprocessing and Aggregation

Before applying LDA to our corpora we processed the documents by doing some preparatory steps such as: removing punctuation and digits, removing whitespace, reducing
the repeated letters (such as the word wooooow), and doing documents tokenization. After that, we aggregate the documents based on their polarity either positive or negative to have two big documents/corpora, one is positive and one is negative. In our experimenter, we have the data sets manually labeled. In addition to the manual labeling, the overall rating for documents can be achieved also by many ways such as utilizing the auxiliary data associated with documents like stars, thumbs, and so forth; and to avoid the gray area between positive and negative, the user can adopt only 5 stars and one star as positive and negative respectively.

Data aggregation was used to address some limitations of LDA such as sparsity and topic incoherency. Short documents like posts used in our experiment are challenging for LDA because of their small length and some informal language used. As mentioned in [92], LDA can learn better and produce coherent topics when provided with long documents. To enrich the model of LDA, short documents are aggregated to give longer documents. Tweets aggregation for example, can be done on hashtags, author’s name, and so on. We did the aggregation by the polarity, positive or negative. The positive corpus consists of 4,030 documents (posts), and the negative corpus consists of 1,545 documents.

4.6 Experiment Results and Discussion

The two main R packages used to implement LDA are topicmodels and lda [93]. LDA package models are all fitted using Gibbs sampling (alternative of direct sampling for high dimensions) for determining the posterior of the hidden (latent) variables. The package tm can be used with variety of files types as the infrastructure of building the corpus through a
number of steps such as: reading the file and transforming the file into document-term matrix (LDA input). The rows of this matrix are the documents names and the columns are the terms. The entry $M_{ij}$ represents the frequency of the jth term in the ith document. So the dimensions of the matrix are the number of documents rows and the number of words columns. The big number of words were reduced dramatically during the preprocessing phase. The preprocessing is a very important step with some critical sub steps like tokenizing (token means word, this exclude other poor data like punctuations or useless words like “the”), removing punctuations and so on. To reduce the size of the matrix, we used tf-idf score (term frequency inverse document frequency) which uses the term-frequency weight to filter unwanted terms according to what we want to achieve (for example, we exclude the terms that appear in most documents because they don’t help to judge the documents). We followed a very useful guide for LDA implementation which can be found on eight2late.wordpress.com website.

\footnote{\url{https://eight2late.wordpress.com/2015/09/29/a-gentle-introduction-to-topic-modeling-using-r/}}
As LDA adopt the assumption Bag of Words (BoW) [82], which neglect the order of words in the document, the backbone of the implementation is Document-Term matrix (dtm). When we get dtm, we apply LDA model on this matrix using Gibbs sampling (set Gibbs values to default). A main step in LDA is to set the number of topics. This is somehow tricky because the topic coherency gets better as the number of topics gets closer to the real number in the corpus. This means when the number of topics is more than what it is supposed to be, the terms will spread over more topics losing their unity under one solid topic, and when the number of topics is less, then more than one topic will be amalgamated under one category (see eight2late.wordpress.com for more details).

To construct our lexicon, we first set the number of topics to two: positive and negative as shown in Figure 4.1. After that, we run LDA over the collected corpus we have with LDA parameters defaults chosen. The two topics means two polarities, positive and negative. As we know, LDA output give the top key terms of each class (topic), where the number of terms is specified by the user in advance. In other words, the term distribution per topic of the whole corpus is drawn from Dirichlet distribution with parameter $\beta$. In our case, the output is two lists (topics) with terms arranged descendingly according to their weight in each topic as shown in Figure 4.4. To make sure the lexicon construction process cover the appropriate range of terms without missing key terms and without adding more cost on the system, we tried a range of number of key terms between 100 and 10,000 testing the accuracy in each point as shown in Figure 4.3.

To test the lexicon, we used it to classify documents collected from Twitter and from
The data sets are the two data sets that we created in previous work [91]. We used manually labeled data as our benchmark to measure the efficiency of our new automatically created lexicon. Data set 1 consists of 5,573 Arabic posts about the Saudi stock market (SSM), 4,029 are positive and 1,544 are negative. Data set 2 contains 2,836 Arabic tweets about SSM as well (2,224 positive tweets and 612 negative tweets). We got good results as shown in Table 4.1. For posts data set, the accuracy and recall reached 92% and 98% respectively; and the accuracy in the case of tweets data set was about 81% and the recall was around 93%.

To do further testing we created new balanced data set consisting of 1,408 tweets (50% positive and 50% negative) using twitter API. For this balanced data set, we got 92%, 72%, and 65% for recall, accuracy, and precision respectively. F1 score was about 77% which is very good when we take in consideration the time and effort we saved by using pLDA (compared to manual lexicon construction). It is worth-mentioning here that although the balanced data is a big challenge for most classifiers, we are still getting high recall (which is

---

2http://www.saudishares.net/vb/
excellent measure compared to accuracy in case of classification).

In the classification process, we adopted the concept of BoW. We treated documents as number of words without considering the relations between words. The documents are classified based on a specific score representing their positivity. The score is the intersection between document and lexicon. So if the positive sentiments in the document are more than the negative sentiments, then the document is classified as positive.

The difference between posts and tweets in terms of length and document nature, has affected the accuracy as shown in Figure 4.3. Tweets are shorter than posts and tend to be more specific (in stock market domain, tweet is sometimes just a word and a number), and as a result, LDA-lexicon is able to classify posts more accurately than tweets. From a very different angle, we can say that LDA-lexicon was built based on a corpus of posts and learned its language much better than the language of tweets although both posts and tweets are in the same general domain which is Saudi stock market. This leads us to what we mentioned before, domain-oriented lexicons outperform general lexicons; and even within one domain, we can build sub-domain lexicon to have better results [73] (for example the domain of laptops can have sub-domain about screen resolution).

4.7 Conclusion

In this chapter, we automatically created Arabic sentiment lexicon for the Saudi stock market using LDA. The beauty of this automated process is simplicity compared to the manual method we used in previous research. The efficiency of LDA-lexicon was high as it showed excellent accuracy and recall for the documents classification task. The only
concern is the overall tag for documents (positive and negative) as it might not be found for some new domains, and this is almost a challenge for most of the learning methods and models. LDA lexicon accuracy showed some fluctuation in the tweets data set but within a very narrow range (less than 2%) while in the case of posts, the accuracy was increasing as the number of key terms increase. It is worth-mentioning that there are some new methods for topic detection that might be tried in future studies. One of those methods is the associative gravity approach (AGA) that was proposed by [94]. The difference between LDA and AGA is that AGA is based only on the content of one single document while LDA is based on corpus. Moreover, the topics are not predefined in AGA as the case in LDA.
5.1 Introduction

Stemming is one of the main steps of text preprocessing. The main goal of stemming is to group all variations of a word in one group to support the process of natural language processing, language modeling, and information retrieval in particular[95]. For example, the word "play" can take several forms according to its function in the sentence, such as the words player, players, playing, played and so on. In some tasks like information retrieval and sentiment analysis, it is very useful to have all those words conflated under one word (stem/root) representing the meaning of the whole set. Stemmers are the algorithms/programs that convert all variations or morphological forms of a word back to the base form. The errors of stemming can be categorized into two main classes: over-stemming and under-stemming. Over-stemming happens when two words grouped together while they should not be, and under-stemming is when two words were assigned to different classes while they should be in the same class[96].

Stemmers are divided into three main types: light stemmers, root-based stemmers, and dictionary-based stemmers. Light stemming can handle most of the variations by cutting prefixes and suffixes. In root-based stemming, the root is extracted first by removing affixes and then the root is found by doing pattern matching process. Dictionary-based stemming
is based on a dictionary of roots and a list of affixes. First, the dictionary is searched to find the word root, and if not found, the word affixes are stripped and the word will be added to the dictionary as a new root. The stemming quality is affected by the dictionary quality[44]. Some other resources categorize stemming approachable into two main parts: language-specific/rule-based approaches and statistical approaches[95].

The most popular English light stemmers are Lovins, Porters, Paice/Husk, and Dawson [97]. Generally speaking, those four stemmers are light(truncating) stemmers which employ affixes lists, conditions, and transformation rules to stem words. The same idea of affixes lists and rules are also in Arabic language as we will see in the following sections. Statistical methods remove affixes using some statistical measures. Examples of statistical stemmers are n-gram, string similarity, and Hidden Markov Model Stemmer (HMM stemmer) which are explained in [97].

In this chapter we introduced a novel Arabic stemmer taking in consideration both statistical methods and language rules adopted by the latest Arabic stemmers such as Khoja [45] and Larkey [51]. Our stemmer is different Khoja stemmer as we do not have to maintain a dictionary of patterns and roots. Compared to Larkey stemmer, our stemmer is equipped with more and different suffixes and we add the enhancement of dealing with infixes.

5.2 Why Arabic Stemmer is Different ?

Stemmers are generally language-specific and there are stemmers for a wide range of languages including Arabic [46]. For Arabic language, stemmers can do a good job to improve effectiveness because Arabic is a highly inflected language. Arabic needs stemming
because of some features of Arabic such as [46]:

- A huge amount of lexical variations caused by orthography and morphology.

- Diacritics represent the vocalization of words, but the challenge is that it is not used all the time which might cause ambiguity and mismatch between vocalized and non-vocalized texts.

- Arabic is highly derivational and inflectional language. In other words, few thousands of roots can produce quite a big number of words by affixes. So, while irregular nouns and verbs are very few in English, it is very frequent in Arabic. For example, the difference between singular and plural is not just simple affixing (we need infixes sometimes, not only prefixes and suffixes).

5.3 Related Work

Good efforts of Arabic stemmers have been done in the last few years. Al-Kharashi and Evens created manual dictionaries of roots and stems[98]. Tim Buckwalter built lexicons of stems and prefixes and suffixes associated with the needed rules to combine stems with affixes[99, 46]. Some Arabic morphological analyzers (roots/grammars) have been done by [100, 101, 102, 103, 45]. Larkey et al. [46] adopted a simple morphological analyzer from Khoja and Garside [45] and did some improvements as shown in Figure 5.1.

Statistical methods have been used as well. Examples of statistical approaches are: the frequency of stems and suffixes were measured by Goldsmith [104], n-grams were used by Oard et al. to find the most frequent final n-grams (1,2,3,4-grams) of words[105]. Mayfield
et al. introduced a system based on word and 6-grams which showed good performance for Arabic and other languages[106]. Al-Fares and De Roeck started by removing some affixes and then they used strings similarity and morphology to cluster words based on their roots [107]. To split up stem classes created by strong stemmers, Xu and Croft used co-occurrence measures to cluster stem classes[108]. Larkey et al. [46] have developed several light stemmers one of them -called light8-s has outperformed Khoja-u (u menas with unbreakables, such as city names) stemmer for both unexpanded and expanded queries. Light8-s did not show that big difference from Khoja (without unbreakables) for unexpanded and expanded queries. Larkey et al. [46] used the software developed by Khoja and Garside to do morphological analysis, and then they used co-occurrence analysis to refine the results.

5.4 Our Approach

Before stemming we did some preparation for the stemming processes and for the input text such as text preprocessing, tokenization, construction of stopwords list (pronouns, prepositions, etc) and construction of other lists such as countries list. Stopwords list consists of 396 words (including country names). In addition to that, we built lists of prefixes and suffixes as shown in Figure 5.1.

5.4.1 Data Set

For this approach, we collected our own data set from two popular arabic newspapers called Alriyadh\textsuperscript{1} and Okaz\textsuperscript{2}. The corpus is extracted from newspapers in random way

\textsuperscript{1}www.alriyadh.com/
\textsuperscript{2}https://www.okaz.com.sa/
Figure 5.1: Larkey Stemmer
covering all areas such as politics, health and sports. For the text crawling, we used OutWit Hub\textsuperscript{3} over one month and we collected about 606K documents (1,704,732 tokens). This data set was used to do some statistics to make the rules of the stemmer phases such as the word length. In addition to that, deep learning utilized this corpus to build lists as we will see shortly.

5.4.2 Lists Construction Using Deep Learning:

Deep learning simplified lists construction as it is an essential part of our stemmer as shown in Figure 5.5. We adopted a deep learning R package called Keras to construct words lists (with some manual filtering) as implemented by Daniel Falbel in the tutorial [109]. Daniel Falbel implemented the skip-gram model created by Mikolov et.al [110]. Skip-gram model is similar to feedforward neural network language model and uses a seed word to predict its neighbors within a predefined window. The input of this model is one-hot representation of the target word, and the output is a vector of size V (vocabulary size) that contains real values (not zeros) of context words as shown in Figure 5.2. A log-linear classifier with continuous projection layer takes the input word and predicts the words within a window it’s center being the current word. The relation between the current word and the words around it gets lower as the distance increases between them. Because of that, Mikolov et.al sample less as the distance between the targeted word and the surrounding words increases. We tried different scenarios and seed words and we found the best results can be achieved by using an algebraic equation [110] of word vectors instead of finding the

\textsuperscript{3}https://www.outwit.com/products/hub/
similar words of only one word. For example, to build stop words list, the following vector equation is much better than using one word only:

\[
Result = embedding\_matrix["\text{اليهودي}\"], drop = FALSE] - embedding\_matrix["\text{الذي}\"], drop = FALSE] + embedding\_matrix["\text{الي التي}\"], drop = FALSE]
\]

Our prefixes list consists of three one-character, four two-character, four three-character, and one four-character prefix terms. The suffixes list contains six one-character, twenty four two-character, fourteen three-character, and ten four-character suffix terms. The stemmer strips prefixes, suffixes, and infixes in a predefined order as mentioned in the phases shown in Figure 5.3.

5.4.3 arStemmer1 Phases:

- Phase1: remove the suffix “تیه” which is a pronoun that indicates the possession as in the example “تداریتیها” her management”, (feminine, singular, third person).

- Phase2: all tokens are checked against the stopwords. Any word match with the stopwords list is skipped without stemming.

- Phase3: detecting the occurrence of four-character prefixes and removing them. The same thing for three-character, two-character, and one-character prefixes in descending order. The condition here is the word must be of length four or more.
Figure 5.2: Skip-Gram Model
Figure 5.3: arStemmer Phases

Table 5.1: Prefixes and Suffixes Lists

<table>
<thead>
<tr>
<th>Pre1</th>
<th>Pre2</th>
<th>Pre3</th>
<th>Pre4</th>
<th>Suff1</th>
<th>Suff2</th>
<th>Suff3</th>
<th>Suff4</th>
</tr>
</thead>
<tbody>
<tr>
<td>و</td>
<td>بال</td>
<td>ال</td>
<td>و</td>
<td>ه</td>
<td>ان</td>
<td>هما</td>
<td>ناهم</td>
</tr>
<tr>
<td>ل</td>
<td>فا</td>
<td>ل</td>
<td>تا</td>
<td>تا</td>
<td>تا</td>
<td>تا</td>
<td>ناهن</td>
</tr>
<tr>
<td>ول</td>
<td>كال</td>
<td>ون</td>
<td>تا</td>
<td>تا</td>
<td>تا</td>
<td>تا</td>
<td>اتها</td>
</tr>
<tr>
<td>اتها</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>اتها</td>
</tr>
<tr>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>اتها</td>
</tr>
<tr>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>كم</td>
<td>نيت</td>
<td>اتها</td>
</tr>
<tr>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
</tr>
<tr>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
</tr>
<tr>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
</tr>
<tr>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
</tr>
<tr>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>وهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
<td>ونهم</td>
</tr>
</tbody>
</table>
• Phase4: remove suffixes in descending order from four-character to one-character suffixes. The word length must be greater than seven, greater than six, greater than five, and greater than four for the suffixes lengths four, three, two, and one respectively.

• Phase5: in this phase we deal with the infixes which is a challenge for most of the stemmers. This is to convert the irregular plural noun to the singular form by removing the letter ”Alif ٠” from the first and fourth locations of the word. For example, the word ”Aqlam أقلام”, which means pens, is stemmed to the singular form ”qalam قلم”.

• Phase6: remove one of the repeated letters at the first position of the word. For example in the word ”Fa-faziaا ففزح”, which means ”and he got scared”, the stemmed word is ”Faziaا ففزح”, which means he got scared.

• Phase7: remove the letter ”waaow و” from words of length four if it was in the third position. This stems some plural words to their singular form like the word ”Oqool / Minds عقول”, stemmed to the word ”Aql / Mind عقل”. In addition to that, this phase stem adjectives and verbal nouns(infinitive or Masdar in Arabic) to the base form as in the word ”Dokhool / Entering دخول”, the stem will be ”Dakhala / he entered دخل”. 

• Phase8: In words of length five, remove the letter ”Taa / T ت” from first position, and the letter ”Yaa / Y ي” from the fourth position. This is to change the infinitive form of the word to the past tense such as stemming the word ”Tatweer / development تطوير”, to the word ”Tawwara / develop تطوير”. 

• Phase9: remove the letter ”Alif / A ٠” from the second position of words of length
four which reduce the infinitive form to the base form (or past tense). For example, we stem the word "Nashir / Publisher ناشر" to the root "Nashara / publish نشر".

5.5 Experiment

We compared our stemmer with the baseline stemmer Khoja as it is very popular and has been cited in many publications.

Khoja stemmer is based on two main steps, preprocessing and matching. In the first step, the text goes through several stages of cleaning by removing diacritics, numbers, punctuation, stop words, the definite article "ال the", the conjunction "و and", and affixation (prefixes and suffixes). In the second step, the word list from step one is matched against a list of patterns and roots, and the match is taken as the root. One additional step is to replace the occurrences of Hamza "ء" with "ئ", and the occurrences of the letters "ي و i" with the letter "و " [47]. There are some weaknesses about Khoja stemmer. The first weakness is the continuous need of maintaining the dictionary as the language changes over time (e.g. new words are borrowed). Secondly, the replacement step with the letter "و" mentioned earlier produce the wrong roots. Third, Khoja failed to remove all affixes [47]. In addition to that, Khoja does not handle proper nouns correctly such as countries names.

We made a direct comparison between our stemmer and Khoja stemmer by giving anonymous copy (the stemmer is not mentioned) of the stemmed words to be evaluated by two native speakers. We did not use the effectiveness of information retrieval as a metric because it is affected by other factors in addition to the stemming process itself, such as database content indexing and query structure.
We used seven Arabic text samples (about 2700 words after removing duplication) taken randomly from different newspapers, and stemmed them by both stemmers Khoja and our stemmer arStemmer1. As shown in Figure 5.4, our stemmer outperformed Khoja in six out of seven tests. To assure the consistency, we created some rules for the comparison:

- Unstemmed word is better than bad stemming word (bad stemming means meaningless word)

- No credit if both stemmers are wrong or both stemmers are equally right.

- If both stemmers are right but with different levels, we take the most correct one. For example, the word "Aljameelah / the beautiful(feminine)الجميلة" can be stemmed into two stems and both are correct: "jameel / beautiful(masculine) جميل", "jameelah / beautiful(feminine) جميلة". In this case, we take the first one as it is simpler.

- Context is used to distinguish between the stem that is still associated with the same class of meaning, and the stem that is deviated to other class or meaning. For example, the Arabic word "Amaliah / practical (feminine)عمليه", has two stems :"amali / practical (masculine) عملي", and "amal / work عمل", we take the first one as it is within the meaning range.

5.6 R Package: arStemmer1

We introduced our stemmer as an R package called arStemmer1. The reason behind that is to be able to attach some data to our solution (prefixes, suffixes, stop words) and to make the stemmer accessible by many users so it can be used and tested as the Stemmer
Figure 5.4: Khoja Vs arStemmer1
can be downloaded from GitHub\textsuperscript{4}. A brief description of the package is shown below:

\textbf{Package: arStemmer1}

\textit{Type: Package}

\textit{Title: Arabic text stemmer}

\textit{Version: 0.1.0}

\textit{Author: Hasan AlShahrani \& Alvis Fong.}

\textit{Maintainer: Hasan AlShahrani <hasan_msh@hotmail.com>}

\textit{Description: The input of this package is Arabic text and the output is the same text stemmed to the base form/roots of words. License: GPL-2}

\textit{Encoding: UTF-8}

\textit{LazyData: true}

\textit{RoxygenNote: 6.1.0}

\textit{Import:}

\textit{tokenizers,stringi}

\textit{Depends: R (>= 2.10)}

\textsuperscript{4}https://github.com/hasan-msh/Rpackage-arStemmer1.git
Figure 5.5: arStemmer1 Main Parts

Figure 5.6: arStemmer1 Algorithm
5.7 Conclusion

We created an R package arStemmer1, which is an Arabic stemmer that takes in Arabic text and return the stems of that text. We tested our stemmer by the direct comparison with the base stemmer Khoja stemmer which is the most popular (and accessible) stemmer for Arabic text. Our stemmer showed good performance through almost all the seven experiments that we did. The stemmer is available for all members of the R community to install and test it.
CHAPTER 6
CONCLUSIONS AND FUTURE WORK

6.1 Summary

In this thesis we introduced a sentiment analysis system for Arabic language focusing on manually building the resources and knowledgeable, automating the process of lexicon construction using topic modeling techniques, and reducing the text dimension by developing a new Arabic stemmer called arStemmer1.

We started by giving an introduction about the Arabic language and why we decided to work with Arabic language instead of using cross languages sentiment analysis. After that, we established our own benchmark by building, labeling, and testing Arabic corpora and lexicons. For the sake of avoiding the hard manual work, we automated the process of lexicon construction using the statistical model Latent Dirichlet Allocation. The last part of this thesis was introducing a new R package called arStemmer1 which an Arabic stemmer available on-line for free use (see chapter 4.7).

Stock market was used as an application of sentiment analysis to test the manually constructed corpora and lexicons in chapter 2.2.8. Investors in the Saudi stock market are used to seeking advice from online resources such as Twitter and discussion forums. We introduced some components to enhance decision making using sentiment analysis and simple fuzzy decision. We built two corpora and one lexicon manually, and we did analysis
on them using both corpus-based approach, and semantical and lexical based approach. The best model was selected to be the base of a fuzzy decision mechanism provided for investors. We mentioned several performance metrics, but the main metric to count on was recall. The best model was the rule-based approach with minimum and maximum recall of 69% and 96% respectively as we go through different data sets, types, and sizes.

Sentiment lexicon is crucial in the process of sentiment analysis. The efficient lexicon is the one that is able to provide the classifier with the right tokens of each class, positive and negative. In this thesis, we have built a domain-oriented Arabic sentiment lexicon automatically using the generative statistical model LDA. We tested our lexicon by doing documents classification and compare it with a classification done based on a manual lexicon created in previous study [19]. We achieved good results from both accuracy and recall perspectives.

Stemming is the processes of removing prefixes, suffixes, and infixes of a word to give the base form of the word. In chapter 4.7, we developed a new stemmer for Arabic language and introduced it as an R package called arStemmer1. We compared our stemmer with the well known stemmer, Khoja stemmer which is one of the best performing stemmers. Our stemmer arStemmer1 outperformed Khoja in six out of seven experiments. We employed deep learning (skip-gram model) to build stop words lists with some manual filtration. The R package arStemmer1 is available for researches to use and test it.
6.2 Future Work

The future work we are interested in is to keep improving the Arabic sentiment lexicon construction to be able to accommodate more Arabic dialects and to have fully automated sentiment analysis system. A good addition might be to use the extensions of LDA or any other topic modeling algorithms to deal with short text documents such as tweets. The stemmer arStemmer1 can be enhanced by more morphological rules and can be augmented by co-occurrence refinement techniques to avoid having two unrelated words in the same stems class.

Moreover, a part of our future work will be the integration of our system with some widely used tools such as R packages or Python libraries.
REFERENCES


Appendix
Algorithm 1 Rules for classification of three classes using the method One-Vs-All.

1: **Input**: Cleansed Corpus of documents, PLA, PLB, NLA, NLB, CL, NegationL
2: **Output**: Annotated corpus.
3: for each document \(d_i\) do
4: \(d_i \leftarrow \text{tokenize} (d_i)\)
5: \(M \leftarrow \text{number of numerical values in } d_i\)
6: \(posA \leftarrow d_i \cap \text{PLA}\)
7: \(posB \leftarrow d_i \cap \text{PLB}\)
8: \(negA \leftarrow d_i \cap \text{NLA}\)
9: \(negB \leftarrow d_i \cap \text{NLB}\)
10: \(comSim \leftarrow d_i \cap \text{CL}\)
11: \(qSim \leftarrow d_i \cap \text{NegationL}\)
12: \(repL \leftarrow \text{either 0 or 1 to represent the absence or presence of repeated letters in } d_i\)
13: end for
14: \(pScore \leftarrow C_1 \times posA + posB\) // positive score
15: \(nScore \leftarrow C_1 \times negA + negB\) // negative score
16: \(SCORE \leftarrow P scorer - C_2 \times N scorer + C_2 \times comSim + C_2/10 \times M - C_2 \times qSim\)
17: if \((SCORE > th1 \lor P scorer > th2) \lor (comSim > 0 \land M > 0)\) then Label = POSITIVE
18: else
19: Label = OTHERWISE
20: end if

Algorithm 2 Classification rules for data set of two classes: positive and negative.

1: In Addition to steps 4-12 in algorithm 1, we do
2: \(pScore \leftarrow 2 \times posA + posB + M + repL\) // positive score
3: \(nScore \leftarrow negA + negB\) // negative score
4: if \((pScore \geq nScore) \lor |d_i| < \text{Threshold1} \land d_i \cap \text{stock list} > \text{Threshold2} \land d_i \cap \text{NLA} == 0\) then LABEL = POSITIVE
5: else
6: LABEL = NEGATIVE
7: end if