

AMERICAN UNIVERSITY OF BEIRUT

SEMANTIC MODELS FOR PRECONCEIVED  
NOTIONS FOR OPINION MINING IN  
ARABIC

by

RAMY GEORGES BALY

A dissertation  
submitted in partial fulfillment of the requirements  
for the degree of Doctor of Philosophy  
to the Department of Electrical and Computer Engineering  
of the Faculty of Engineering and Architecture  
at the American University of Beirut

Beirut, Lebanon  
November 2016

# AMERICAN UNIVERSITY OF BEIRUT

## SEMANTIC MODELS FOR PRECONCEIVED NOTIONS FOR OPINION MINING IN ARABIC

by  
RAMY GEORGES BALY

Approved by:

Dr. Ayman Kayssi, Professor

Department of Electrical and Computer Engineering



Chair of Committee

Dr. Hazem Hajj, Associate Professor

Department of Electrical and Computer Engineering



Advisor

Dr. Hassan Artail, Professor

Department of Electrical and Computer Engineering



Member of Committee

Dr. Wassim El-Hajj, Associate Professor

Department of Computer Science



Member of Committee


Dr. Kathleen Mckeown, Professor  
Columbia University



Member of Committee

---

Dr. Nizar Habash, Associate Professor  
New York University, Abu Dhabi



Member of Committee

---

Dr. Khaled Bashir Shaban, Associate Professor  
Qatar University



Member of Committee

Date of thesis defense: November 1, 2016

# AMERICAN UNIVERSITY OF BEIRUT

## THESIS, DISSERTATION, PROJECT RELEASE FORM

Student Name: Baly Ramy Georges  
Last First Middle

Master's Thesis       Master's Project       Doctoral Dissertation

I authorize the American University of Beirut to: (a) reproduce hard or electronic copies of my thesis, dissertation, or project; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes.

I authorize the American University of Beirut, to: (a) reproduce hard or electronic copies of it; (b) include such copies in the archives and digital repositories of the University; and (c) make freely available such copies to third parties for research or educational purposes after: **One \_\_\_ year from the date of submission of my thesis, dissertation or project.**

**Two \_\_\_ years from the date of submission of my thesis , dissertation or project.**

**Three \_\_\_ years from the date of submission of my thesis , dissertation or project.**

[Signature]  
Signature

January 9, 2017  
Date

# Acknowledgements

Firstly, I would like to express my sincere gratitude to my advisor Prof. Hazem Hajj for his motivation, knowledge, and continuous support, both at the academic and personal level, since day one of my graduate school marathon. His guidance was of immense help for me throughout this venture, and was crucial to the realization of this thesis. I could not have imagined having a better advisor and mentor for my PhD.

I have also been fortunate to work under the supervision of outstanding scholars: Prof. Nizar Habash (New York Abu Dhabi), Prof. Wassim El-Hajj (AUB) and Prof. Khaled Shaban (Qatar University) who were generous to provide me with guidance and insightful comments that incited me to widen my research from various perspectives. I am also thankful to all former and present members in the OMA research team who helped me a lot through collaborations and discussions.

Finally, and most importantly, I would like to thank my beloved wife Ruby for her support, encouragement, patience and tolerance, especially in patches of rough times. Her unwavering love was the bedrock upon which the past seven years of my life have been built. I also thank my parents, Georges and Caroline, for their moral support and their faith in me and allowing me to be as ambitious as I wanted.

# An Abstract of the Dissertation of

Ramy Georges Baly for Doctor of Philosophy  
Major: Electrical and Computer Engineering

Title: Semantic Models for Preconceived Notions for Opinion Mining in Arabic

Opinion mining (OM), or sentiment analysis (SA), is the process of having computing machines automatically understand and interpret text to identify opinions expressed on certain subjects. OM has become interesting given the abundance of user-generated opinionated data on the Web, especially on social media websites (Facebook, Twitter, LinkedIn, etc.). OM has significant applications in Politics, Social Media, and Business. Providing insights into the public opinion shapes critical decisions in these fields such as influencing voters' directions in the currently active US elections, or having a business decide on a new product launch.

OM belongs to a multi-disciplinary research intersecting the fields of machine learning, psychology, social media, and natural language processing (NLP), with NLP presenting the most significant challenge in having machines automatically understand the semantics in text. In this thesis, we explore solutions for opinion mining in Arabic (OMA) due to Arabic's importance as the 5th most-spoken language worldwide, and that recently became a key source of internet content with a 6,600% growth in number of users compared to the year 2000.

OMA's challenges include Arabic's lexical sparsity and ambiguity due to its rich morphology, where Arabic words are packed with significant amounts of information through complex concatenative and inflectional systems. Arabic has also a wide range of dialectal variations, as well as ambiguity caused by optional diacritization in Arabic scripts. Additionally, Arabic suffers from the lack of reliable large-scale sentiment lexical resources that can help training and evaluating accurate machine learning models.

To address these challenges, we present solutions inspired from Psychology and Neuroscience. We present a meta-framework to automate the human cognitive processes while reading and interpreting sentiment. Furthermore, for the inference step, we develop new deep learning methods for OMA, inspired from

neuroscience and state-of-the-art neural network algorithms that aim at representing the brain's neurons and their activations. Experiments with state of the art methods in English and Arabic show the superiority of our methods.

# Contents

<b>Acknowledgements</b>	<b>v</b>
<b>Abstract</b>	<b>vi</b>
<b>Contents</b>	<b>viii</b>
<b>List of Figures</b>	<b>x</b>
<b>List of Tables</b>	<b>xi</b>
<b>1 Introduction</b>	<b>1</b>
<b>2 Related Work</b>	<b>6</b>
2.1 Opinion Mining in English . . . . .	6
2.1.1 Feature Engineering-based Models . . . . .	6
2.1.2 Deep Learning-based Models . . . . .	8
2.2 Opinion Mining in Arabic . . . . .	10
<b>3 Challenges and Objectives</b>	<b>13</b>
3.1 Challenges Related to Opinion Mining Meta-Framework . . . . .	13
3.2 Challenges in OMA . . . . .	14
3.2.1 Overview of Arabic Morphology . . . . .	14
3.2.2 Arabic-specific Challenges . . . . .	15
3.3 Objectives . . . . .	18
<b>4 Human Reading for Opinion Mining</b>	<b>20</b>
4.1 Models of Human Reading . . . . .	20
4.2 Research gaps with respect to HRS . . . . .	23
4.3 Applications of the HRS Meta-Framework . . . . .	26
4.3.1 HRS with Feature Engineering: FRN as Case Study . . . . .	26
4.3.2 HRS with Deep Learning: GRNN as Case Study . . . . .	31
4.4 Experiments and Results . . . . .	35
4.4.1 Datasets and Evaluation . . . . .	35
4.4.2 Evaluating FRN with HRS . . . . .	36



4.4.3	Evaluating GRNN with HRS . . . . .	39
<b>5</b>	<b>Recursive Auto Encoders for OMA</b>	<b>43</b>
5.1	RAE for Opinion Mining . . . . .	43
5.2	RAE for Opinion Mining in Arabic . . . . .	46
5.2.1	Tokenization to address Sparsity and Ambiguity . . . . .	46
5.2.2	Sentiment Embedding to improve Word Embeddings . . . . .	47
5.2.3	Syntactic Parsing to improve Composition . . . . .	50
5.3	Experiments and Results . . . . .	50
5.3.1	Datasets and Experimental setup . . . . .	50
5.3.2	Ablation Analysis . . . . .	52
5.3.3	Results Benchmarking . . . . .	55
<b>6</b>	<b>Morphologically-Enriched Recursive Deep Models</b>	<b>58</b>
6.1	RNTN for Opinion Mining . . . . .	58
6.2	Addressing Rich Morphology & Ambiguity . . . . .	60
6.3	The Arabic Sentiment Treebank . . . . .	61
6.3.1	System Architecture . . . . .	61
6.3.2	Description and Intrinsic Evaluation . . . . .	63
6.4	Experiments and Results . . . . .	66
6.4.1	Experimental Setting . . . . .	67
6.4.2	Results . . . . .	67
<b>7</b>	<b>Conclusion</b>	<b>75</b>
<b>A</b>	<b>Tools and Software</b>	<b>77</b>
	<b>Bibliography</b>	<b>80</b>

# List of Figures

4.1	The human reading process. . . . .	22
4.2	Application of the HRS meta-framework to the FRN method. . .	27
4.3	The architecture of GRNN with and without HRS. (a) shows the original GRNN framework. (b) shows the modified GRNN framework with the modeling of the “ <i>situation model of interpretation</i> ”	34
4.4	Evaluating different sizes of the domain lexicon. . . . .	37
4.5	Evaluating the impact of synonymy when introducing SYN-phrases vs. phrases under different sizes of the domain lexicon. Figure (a) shows the accuracies, while Figure (b) shows the F1 scores. . . . .	38
4.6	Evaluating the FRN method with and without HRS under different sizes of reduced features. (a) shows results on the IMDB corpus. (b) shows results on the YELP corpus. . . . .	40
5.1	The framework of RAE, which we refer to as baseline RAE. . . .	44
5.2	The structure of the AE (parameters, inputs and outputs). . . .	45
5.3	The framework of RAE for opinion mining in Arabic, with solutions highlighted in blue. . . . .	47
5.4	The architecture of the C&W embedding model, extended with pre-training stage. . . . .	49
6.1	Sketch of RNTN applied to predict sentiment in a three-word phrase.	59
6.2	Architecture of the system to develop ARSENTB. . . . .	62
6.3	The different morphological features provided in ARSENTB . . . .	64
6.4	Normalized distribution of aggregated sentiment labels at each level of $n$ -gram in ARSENTB. . . . .	65

# List of Tables

1.1	Impact of Arabic complex morphology on sentiment. . . . .	4
4.1	Qualitative analysis of document-level sentiment analysis models against aspects of human reading. . . . .	25
4.2	Characteristics of datasets used in experiments. . . . .	36
4.3	Results obtained on both datasets (YELP with 5 classes and IMDB with 2 classes) using GRNN with and without HRS. . . . .	41
4.4	Summary of all results obtained by applying HRS to FRN and GRNN approaches on the IMDB corpus (5 classes) and the YELP corpus (5 classes) . . . . .	42
5.1	Characteristics of the different evaluation corpora. . . . .	51
5.2	Impact of each solution evaluated on the ATB corpus. . . . .	53
5.3	Impact of each solution evaluated on the QALB corpus. . . . .	53
5.4	Impact of each solution, evaluated on the Tweets corpus. . . . .	54
5.5	Impact of different tree structures on the performance of RAE for opinion mining in Arabic. . . . .	55
5.6	Results of benchmarking the performance of RAE for OMA against sentiment models from the literature. . . . .	57
6.1	Sentiment distribution in ARSENTB . . . . .	65
6.2	Different types of sentiment compositions observed in ARSENTB . . . . .	66
6.3	Evaluation splits of ARSENTB . . . . .	67
6.4	The impact of adding orthographic features on the performance of baseline RNTN for five-way classification. . . . .	68
6.5	The impact of adding orthographic features on the performance of baseline RNTN for three-way classification. . . . .	69
6.6	Performance of the different RNTN models, and comparison to a variety of SVM classifiers. . . . .	70
6.7	The impact of adding morphological features on RNTN performance in English. . . . .	71
6.8	A qualitative analysis of the recursive deep models for OMA against aspects of human reading. . . . .	73

A.1	Software and tools required to train and evaluate the applicability of the HRS meta-framework to the “FRN” opinion model . . . . .	77
A.2	Software and tools required to train and evaluate the applicability of the HRS meta-framework to the “GRNN” opinion model . . . . .	78
A.3	Software and tools required to train and evaluate the “RAE” model for OMA . . . . .	78
A.4	Software and tools required to train and evaluate the “RAE” model for OMA . . . . .	79

# Chapter 1

## Introduction

Opinion mining (OM), or sentiment analysis (SA), are used interchangeably to refer to the task of the automatic identification and extraction of opinions or sentiments expressed towards targets (individuals, organizations, events or topics) in textual data. With the rapid growth of the Web, huge amounts of opinionated data are flooding the Internet in social media websites (Facebook, Twitter, Instagram, etc.), forums, blogs and reviews websites [1]. For instance, Facebook servers handle more than 30 Petabytes of data including text, images and videos. Also, more than 230 million tweets are tweeted on a daily basis <sup>1</sup>.

Developing opinion mining models that are capable of analyzing and understanding the subjective content of the Web has become a valuable asset for business intelligence, marketing, politics and a wide range of domains. It provides companies with insights into costumers' opinions about their services, and also provides individuals with summarized reviews and opinions about items they would like to buy. It can be used to predict stock market prices, and also to model the people's behavior during major events. For example, the death of *Steve Jobs* caused a sad sentiment spike on Twitter [2]. It was also used at the onset of the 2012 US presidential elections to track public popularity [3]. The diversity of sentiment-related applications and the benefits of getting access into people's opinions have fueled research on sentiment analysis over the last two decades.

Sentiment analysis encompasses several sub-tasks [4]. *Sentiment classification* is one of the most researched sentiment-related tasks [5] that aims to determine the overall sentiment polarity of text ranging from single words to full documents. It is particularly useful for analyzing customer reviews, which usually target a single entity (item, product or service). *Aspect-based sentiment classification* aims to identify the different sentiments expressed towards aspects of target entities, providing richer and deeper results [6]. *Aspect-based opinion summarization* aims to provide a comprehensive and summarized representation for a large num-

---

<sup>1</sup><http://www.ibmbigdatahub.com>

ber of opinions and sentiments according to the different aspects that are being evaluated [7]. *Identifying implied sentiment* is a challenging sentiment analysis task that aims to analyze objective sentences that are used to express subjective opinions. This task requires common-sense and knowledge of the application domain [8]. At last, *co-reference resolution* aims to discover mentions of entities and aspects of these entities, and relate them to each other [9]. This dissertation is primarily focused on developing accurate models for sentiment classification, where given a text segment, the aim is to predict its overall sentiment (whether it is positive, negative or neutral) and the strength of that sentiment.

In this dissertation, we target two objectives. The first is to develop a meta-framework for opinion mining that mimic the human’s reading process. The second is to advance state-of-the-art inference models for opinion mining in Arabic.

Research in opinion mining has been driven by the need to improve the accuracy performance of existing models. Although models have already achieved state-of-the-art performances in English, we conjecture that best accuracies can still be achieved by models that come closest to the human reading. Therefore, we take a fundamental approach in attempting to match the human reading process for the purpose of sentiment classification. This approach builds its foundations from the field of psychology by understanding and automating the cognitive process that humans follow to read and interpret sentiment from text.

Based on the human reading theory in psychology [10], we formulate sentiment analysis in a human-like “meta-framework”, and we refer to it as the “Human Reading for Sentiment” (HRS). A central idea to HRS is the capture of notions that include both textual representations and sentiment polarities [11]. We provide a comprehensive coverage of the “notion” concept by including an expanded definition and proposing additional human-related features such as the working memory and the notions synonyms. The resulting HRS meta-framework enables the development of opinion mining models that are complete in their representation of the human reading process. It can be used to extend existing models, or as a reference when developing new models from scratch.

In this dissertation, we use HRS to extend two state-of-the-art methods for document classification, originating from two schools of machine learning. The first method is based on feature engineering and uses feature relation networks (FRN) to reduce a rich set of heterogeneous  $n$ -gram features [12]. The second method is based on deep learning and uses Gated Recurrent Neural Networks (GRNN) [13]. Although both models already achieve very high accuracies on different reviews datasets, experimental results showed that extending these models according to the HRS yields further improvements.

Towards the objective of advancing inference models for opinion mining, a wide range of models were previously developed based on feature engineering, where different types of features were extracted to train machine learning models. Each type of features aims to capture a specific characteristic of the text,

including the shallow surface of words and their context [14,15], the syntactic rules that define the structure of the text and the relations between its words [16–18], and the semantics of the text including meanings and sentiments of words or phrases it contains [12, 19, 20]. The extraction of these features uses advanced natural language processing (NLP) tools such as parsers, stemmers, and taggers, as well as large-scale lexical resources such as sentiment lexicons [21] and semantic thesauri [22].

Current state-of-the-art models for opinion mining are based on deep learning techniques. These models have achieved high performances coupled with eliminating the need for labor-intensive feature engineering. Deep learning has recently emerged as a major advance in machine learning and artificial intelligence. This class of models aims to simulate the way the human brain works by mimicking activities in several layers of neurons, where each of these layers extracts some latent feature or aspect of the task they are trained for, using raw unlabeled datasets [23]. This technology has become possible due to the current increase in computational and processing capabilities, as well as the abundance of raw unlabeled datasets such as the Wikipedia textual corpus [24] and the ImageNet database [25]. As a result, deep learning was successfully used in a wide range of applications such as image recognition [26], machine translation [27] and text and sound generation [28,29]. Most deep learning models for opinion mining are based on the concept of compositionality [30]. To perform semantic composition, neural language models (NLM) are used to generate word embeddings that capture distributional semantic and syntactic aspects of each word [31]. These embeddings are then combined together, through some composition function, to derive an overall representation (or embedding) of the text that is used to predict its sentiment. Semantic composition models can be categorized into sequential models that cumulate the information through the text [13, 32] and recursive models that propagate the information up a parse tree [33,34]. Overall, recursive models have proved more effective, mainly because the meaning in a language is also constructed recursively [35].

In this thesis, we place special focus on developing inference models for opinion mining in Arabic (OMA) since Arabic language is the 5th most-spoken language in the world [36], and has recently become a major source of the Internet content with a 6,600% growth in number of users compared to the year 2000 [37]. Performing OMA is a timely and intriguing application that is worth investigating.

Research on OMA has been following the footsteps of opinion mining research in English, mainly focusing on feature engineering to train machine learning models [38–40]. However, it is faced with several challenges related to the language complexity, the relative recency of Arabic NLP, and the limited semantic resources. Arabic is a morphologically-rich language (MRL), where words contain many morphemes and are packed with significant amount of information through a complex concatenative and inflectional system, leading to lexical sparsity. Furthermore, the fact that diacritization is optional in Arabic scripts

leads to lexical ambiguity. Table 1.1 shows an example of how an Arabic token can have different tokenizations and diacritics (عَلَامَاتُ التَّشْكِيلِ), producing different words with different meanings and sentiments. This example also shows the “information-packing” aspect, where a single Arabic word corresponds to different multi-word English phrases. Furthermore, OMA was mostly performed at the level of sentences and documents, whereas opinion is often expressed in more subtle manners and needs to be modeled at lower levels of granularity including words and phrases. Research in this direction is held back by the lack of corpora with fine-grained annotations, which are expensive and time-consuming to create. Moreover, despite the recent release of several lexical resources including sentiment lexicons [41, 42] and annotated corpora [38, 39], the number and diversity of these resources are still small compared to those developed in English. Finally, a large number of non-standardized dialectal variants exist for Arabic and are excessively used by Internet users instead of Modern Standard Arabic (MSA). This adds to the complexity of Arabic NLP tools that are needed to process and analyze colloquial text. Recent efforts have been achieved for Egyptian dialect [43, 44].

Surface	وبشرها <i>wb\$rhA</i>		
Tokenization	و+بشر+ها <i>w+b\$rh+hA</i>	و+ب+شر+ها <i>w+b+\$r+hA</i>	و+بشر+ها <i>w+b\$rh+hA</i>
Diacritics	وَبَشْرَهَا <i>waba\$~arahaA</i>	وَبِشْرَهَا <i>wabi\$ar~ihaA</i>	وَبَشْرَهَا <i>waba\$arihaA</i>
Gloss	‘and he told her good news’	‘and with her evil’	‘and her people’
Sentiment	<b>Positive</b>	<b>Negative</b>	<b>Neutral</b>

Table 1.1: Impact of Arabic complex morphology on sentiment.

To address the above-mentioned challenges in Arabic, we propose to use deep learning, which has not yet been explored for OMA. Deep learning models used in English can not be directly applied to Arabic due to several reasons. 1) The ability of deep learning to capture syntactic and semantic aspects of the words from raw text will be affected by the morphological complexity and ambiguity of Arabic language. 2) Given Arabic’s rich concatenative morphology, raw words usually contain several morphemes and can be tokenized into different levels of morphological detail. Therefore, training recursive deep models with raw words will neglect the morpheme-level semantic interactions that may contribute to the overall meaning. 3) The composition models are trained using word embeddings



that do not capture sentiment aspects of the words. This affects the ability of the composed embeddings to predict the sentiment of the text.

To overcome these limitations in Arabic, we propose 1) to improve the word embeddings by incorporating sentiment into the word vectors, 2) to improve path of recursion of composition by using syntactic rules of the language, and 3) to enrich the composition models with morphological and orthographic features in order to abstract away from the sparse raw words and reduce ambiguity. Furthermore, we present ARSENTB the first Arabic sentiment treebank that supports recursive deep models for Arabic sentiment analysis at different levels of text starting from the word-level. The resulting enhanced models achieved significant performance improvements of around 10% in both accuracy and F1-score, on several datasets, compared to the baseline models. They also outperformed well-known classifiers trained with similar morphological considerations, highlighting the advantage of using deep learning for opinion mining.

The remaining of this dissertation is organized as follows. Chapter 2 describes previous efforts done for opinion mining in both English and Arabic languages. Chapter 3 describes the main challenges associated with OMA, and summarizes the thesis objectives. Chapter 4 presents the psychological foundation of the “human reading for sentiment” (HRS) meta-framework, describes the proposed steps for HRS automation, and shows improvements over several state-of-the-art models. Chapters 5 and 6 describe the solution models to advance inference models for opinion mining in Arabic, with focus on using recursive deep models. Finally, conclusions are presented in Chapter 7.

# Chapter 2

## Related Work

In this chapter, we provide an overview of previous opinion mining methods. Since significant efforts have been made for English language, We start by describing methods proposed for English opinion mining, since research on OM has been established for that language. Then, we describe previous models proposed for Arabic language.

### 2.1 Opinion Mining in English

Methods for opinion mining in English can be categorized into in two groups. The first group is based on feature engineering to train classical machine learning models. The second is based on recent advances in deep learning.

#### 2.1.1 Feature Engineering-based Models

Given the availability of advanced NLP tools as well as large-scale semantic and sentiment resources, a wide range of feature types were made available through feature engineering to train machine learning models for opinion mining, achieving high accuracies. These features vary in the complexity of the information they capture, ranging from shallow surface, to syntactic and deeper semantics.

Surface features (bag-of-words (BoW) or word  $n$ -grams) were the earliest features to be explored for sentiment classification. Word  $n$ -grams were used to train several classifiers such as Support Vector Machines (SVM), Naïve Bayes (NB) and Expectation Maximization (EM) to perform opinion mining in movie reviews [14]. They were also used to identify hidden sentiment factors that helped improving product sales prediction [45]. Results in [15] showed that extracting the word  $n$ -gram features from the subjective parts of the document only achieved better performances. In general, surface features provide a shallow insufficient representation of the text semantics.

Research in NLP has led to the development of tools that can accurately

apply tokenization, stemming, lemmatization, part-of-speech (POS) tagging, dependency and constituency parsing, and many other NLP tasks. Examples of such tools include the Stanford CoreNLP [46] and the Natural Language Toolkit (NLTK) [47]. Such NLP advancements have made it possible to incorporate syntactic information that reflect the principles by which language is constructed as additional features to train the sentiment analysis models.

Predefined POS tag patterns were used to extract subjective phrases from text. Then each phrase was assigned a sentiment score using its point-wise mutual information (PMI) with highly-subjective keywords, and scores are used to perform unsupervised sentiment classification in consumer reviews [16]. Stemming was used to generalize and reduce sparsity of the Multi-Perspective Question Answering (MPQA) and the appraisal lexicons [48, 49], which are then used to perform unsupervised sentiment classification in different reviews datasets [50]. Modification relations extracted from dependency parse trees were used along with POS tags for phrase-level sentiment classification [17]. A collection of syntactic (word and POS  $n$ -grams) and stylistic features (letter frequencies, character and digits  $n$ -grams, function words, and word length) achieved high sentiment classification accuracies in web forum content, after being reduced by the Entropy-weighted genetic algorithm (EWGA) [18]. Experiments showed that syntactic features achieved better performance than stylistic features, and that the union of both achieved the highest performance.

A deeper understanding of the text semantics was incorporated into the models through the development of several semantic and sentiment lexical resources. Sentiment lexicons are dictionaries of words or phrases associated with sentiment indicators (either scores or labels), and are developed through manually [48], automatically [51] or semi-supervisedly [21]. Examples of sentiment lexicons include the General Inquirer (GI) [52], MPQA [48], Bing Lu lexicon [7], SentiWordNet [21], NRC-hashtag [53] and Sentiment140 [54]. Other lexical resources include databases of words and phrases with richer semantic information such as synonyms, hypernyms, hyponyms, Is-A, assertions and affect. Examples of such resources include WordNet [22], ConceptNet [55] and the Dictionary of Affect in Language (DAL) [56]. As a result, the above-mentioned resources allowed to extract a variety of semantic and sentiment features. Most of these features are centered around the word-level sentiment, and are used either alone or with other features to train sentiment analysis models.

Word-level sentiment polarities, extracted from the GI lexicon and extended by the WordNet synonym relations, were used with dependency relations for sentiment classification at the phrase-level [19]. Word-level sentiment scores were calculated from the dictionary of affect (DAL), along with BoW and syntactic features, were also used for phrase-level sentiment classification [57]. Subjective words from the MPQA lexicon were used to identify the subjective syntactic sub-structures in parse trees, from which BoW features were extracted to perform sentiment classifiers in movie reviews [58]. The SentiWordNet lexicon has

a special structure, where each synset in WordNet is assigned three scores indicating the positivity, negativity and neutrality of words in that synset [21]. SentiWordNet was used to extract sentiment score  $n$ -grams [20], and also to extract sentence-level aggregate features such as the number, type and polarity score of words and phrases [59]. WordNet was used to generalize the semantics of word  $n$ -grams, by replacing each word with its synset label. These features were extracted as part of a rich set of heterogeneous features including surface features (character and word  $n$ -grams) and syntactic features (POS, word-POS  $n$ -grams and subjective phrase patterns). Due to the huge size of the proposed feature set is huge in size, with many redundant or irrelevant features, feature relation networks (FRN) was proposed to produce a smaller and more representative feature set that was able to achieve high sentiment classification accuracies on product and movie reviews [12]. Several sentiment lexicons were combined to increase the coverage of extracting word-level sentiment scores, that were used along with a set of hand-crafted stylistic features (count of all-caps words, emoticons and location, frequency of elongated words, count of negations, punctuation, etc.) to train the ‘NRC’ model that won the SemEval-2014 shared task on contextual polarity classification [53]. Concatenating these features with sentiment-specific word vector representations (embeddings) achieved further performance improvement [60]. Finally, a preprocessing framework that includes normalization, spelling correction, POS tagging, word sense disambiguation and negation detection was used to extract character and word  $n$ -grams, word-level scores from sentiment lexicons, and the different senses for each word. The resulting model was the winner of the SemEval-2014 shared task on message-level sentiment classification [61].

### 2.1.2 Deep Learning-based Models

Deep learning has recently emerged as a major advance in machine learning. It aims to simulate the highly-complex neural structure of the human brain by using several layers of neurons to model complex non-linearities that help accomplishing complex tasks [23].

Although features extracted using feature engineering can be used as input to deep learning models, this is not recommended due to their suffering from high-dimensionality, high degree of sparsity, and assume independence among features with lack of word order, which affect the model’s ability to infer useful patterns from the input data. Alternatively, deep learning models are trained using input word vectors (or word embeddings) that are derived using vector space models (VSM). Common VSMs are based on matrix factorization techniques such as Latent Semantic Analysis (LSA) [62] and Latent Dirichlet Allocation (LDA) [63]. Currently, neural language models (NLM) are used to generate the word embeddings using deep neural networks that take random word vectors as input and fine-tunes them by back-propagating the error of predicting their local context in a large unlabeled corpus. The resulting vectors capture distributional

semantic and syntactic aspects of each word, based on its co-occurrence statistics information in a large corpus [31]. Currently-used NLMs include the validity-based C&W model and the probabilistic Word2vec model [64] that utilizes the skip-gram or the continuous bag-of-words (CBOW) architectures.

Following their success in NLP, several deep learning approaches were successfully used to perform sentiment analysis, by learning embedded semantic representations of the text that can be used to perform accurate sentiment classification. Stacked Denoising Auto Encoders were trained using word  $n$ -gram features for sentiment prediction in a large corpus of Amazon reviews [65]. A neural language model (NLM) was extended to derive embeddings of longer span of text (up to paragraphs), such that the resulting vectors capture the semantics distributed over all words in that text [66]. This approach achieved high results when used to classify sentiment in movie reviews. Current state-of-the-art deep learning models for sentiment are based on the concept of compositionality; the meaning of a compound text can be described as a function of 1) the meanings of its parts (constituents) and 2) the rules by which these parts are combined [30]. Compositionality was mainly used to derive the meaning of sentences, and therefore composition models can be seen as a function that transforms a sequence of word embeddings into a sentence embedding that captures the semantic and syntactic interactions among its words, and that is used to train the sentiment classifier, which is usually a logistic regression “softmax” layer. Several composition models have been proposed, and these can be categorized into sequential and recursive. Sequential models accumulate information over each sentence sequentially. At each iteration of composition, the model derives an embedding that captures interactions between the current word and all preceding words in the sentence. This process continues until the model reaches the end of the sentence. Examples of sequential models include Recurrent Neural Networks (RNN) [32] and Convolutional Neural Networks (CNN) [67,68]. On the other hand, recursive models propagate the information up a binary parse tree. At each iteration of composition, the model derives an embedding that captures interactions between two constituents (words or multi-word phrases) selected based on the structure of a parse tree. This process continues until the model reaches the root node of the tree. Examples of recursive models include Deep Recursive Neural Networks (DRNN) [69], Recursive Auto Encoders (RAE) [33], Recursive Neural Tensor Networks [34] and Tree-structured Long Short Term Memory (LSTM) [70]. Most of the aforementioned deep models perform composition in a totally unsupervised way, and sentiment classification that takes place on top of the composition is the only supervised task in the model. Overall, recursive models proved more effective than sequential models, mainly because the meaning in a language is also constructed recursively [35]. Recursive models can also perform sentiment classification at the word and the phrase-level. Results in [34] proved that using fine-grained sentiment to model composition was able to improve the accuracy of sentiment classification at the sentence-level. However, this requires a sentiment

treebank; a collection of parse trees annotated for sentiment at all levels of constituency. The only such treebank that exists so far is the “Stanford Sentiment Treebank” [34]. For document sentiment classification, a document composition function was added on top of the existing sentence composition. This function derives a document-level embedding that captures relations and interactions between its sentences. Gated Recurrent Neural Networks (GRNN) were proposed for document composition due to their ability to model long-distance dependencies between sentences [13].

## 2.2 Opinion Mining in Arabic

Methods for OMA are mostly based on feature engineering to train machine learning models. The feature space that has been explored in Arabic is similar to that in English, except for features proposed to address Arabic-specific issues, such as the morphological complexity of the language.

Surface features, represented by word  $n$ -grams, were extensively evaluated under different settings including different lengths of context  $n$ , different pre-processing (raw words vs. stemming) and different feature representation scores (presence, term frequency ‘TF’ and term frequency inverse document frequency ‘TFiDF’). Word *bi*-grams and *tri*-grams achieved best results when used to train Support Vector Machines (SVM) [38, 71–73]. Naïve Bayes (NB), on the other hand, achieved competitive performances [74, 75], and ensemble classifiers achieved further improvements [76]. Stylistic features (letter and digit  $n$ -grams, word length, etc.) were proposed with surface features, and achieved high performances after being reduced using the Entropy-Weighted Genetic Algorithm (EWGA) [18]

Arabic is well-known for its morphological richness and complexity. Hence, it is of high importance to incorporate syntactic and morphological information of the language into the sentiment models. One of the earliest grammatical approaches proposed to generalize verbal and nominal phrases into one form based on ‘actors’ and ‘actions’, and then to train SVM using the following features: actors, actions, adjectives, nouns, syntactic type of sentence, conjunction with previous sentence, and word sentiment polarity [77]. Most of this information was manually labeled due to the lack of Arabic NLP tools at the time. Results are considered as a “proof of concept” of the importance of incorporating syntactic information into the model.

The recent establishment of advanced Arabic NLP tools and resources allowed the automatic extraction of syntactic and morphological features that aim to mitigate the impact of complex morphology on sentiment. Examples of such resources include the Arabic Treebank (ATB) [78], SAMA for morphological analysis [79], MADAMIRA for morphological analysis and disambiguation [43]. For instance, adding word-level inflectional morphology (gender, number, voice, ...) to basic surface features improved the performance when applied to data written

in MSA [80]. However, these features caused performance degradation when applied to tweets [81]. The main reason for this observation is that most Arabic NLP tools are trained on MSA data, whereas Twitter data contain significant amounts of dialects and misspellings, causing the extraction of inaccurate features. Using stem and lemma  $n$ -grams achieved better results than using word  $n$ -grams, reflecting their ability to generalize lexical variations of Arabic words [40, 80]. However, it is not clear, which feature is better to use. The impact of adding part-of-speech (POS) tags to lemma  $n$ -grams was evaluated for both subjectivity and sentiment analysis tasks, where subjectivity analysis determines whether a text is subjective or objective [40]. Adding POS information did not improve sentiment classification, while it slightly improved subjectivity classification.

In addition to syntax, it is crucial to equip sentiment models with deeper insight into the text semantics. This kind of information was made possible after the development of several sentiment lexicons, that we briefly describe next. ARABSENTI contains 3,982 adjectives extracted from 400 documents belonging to the ATB Part 1 V3.0 [78]. These adjectives were manually labeled as positive, negative or neutral [80]. ARSELEX is an automatically-generated lexicon that contains 5,244 adjectives that are expanded from a gold set of 400 adjectives [82]. ARSENL is a large-scale sentiment lexicon that contains 28,760 entries, represented by their lemmas, each associated with three scores indicating the positivity, negativity and neutrality of the corresponding entry [42]. This lexicon was created by linking between different lexical resources including the English WordNet [22], the Arabic WordNet [83], SentiWordNet [21] and the Standard Arabic Morphological Analyzer (SAMA) [79]. Standard Arabic Sentiment Lexicon (SLSA) contains 34,821 entries, and was created following the same approach used to create ARSENL, with additional heuristics and back-off strategies to increase coverage [84]. Finally, SANA contains nearly 240K entries that cover words in both standard and dialectal Arabic [41]. It is composed of different lexicons, some were annotated manually, while others were automatically-translated from English lexicons such as SentiWordNet [21], the General Inquirer [52], the Affect Control Theory (ACT) lexicon [85], in addition to frequent terms in the Youtube dataset [86].

These and many other sentiment lexicons were used to support unsupervised sentiment analysis models that aggregate word-level sentiments to determine the overall sentiment of a sentence [87, 88]. They also helped training machine learning models with different types of semantic features. For instance, word-level sentiment scores extracted from ARSENL were averaged across all words of a sentence, and then used to train sentiment classification models [42, 89, 90]. ARABSENTI and SANA were used to create binary features that indicate the presence of subjective, positive and negative adjectives in the text. These features helped improving classification performance, even when added to complex feature sets [41, 80–82]. Sentiment score *uni*-grams were used with word  $n$ -grams and POS tags for sentence-level classification [91]. Experimental results indicate that

extracting the scores from ARABSENTI, extended through graph reinforcement, achieved similar performances to extracting scores from a translated version of the English MPQA lexicon.

Given that extensive work have been already done for sentiment analysis in English, and has achieved high levels of accuracy, machine translation (MT) was explored to perform sentiment analysis in less-researched language, including Arabic. In this framework, text from the original language is automatically translated to English, and then state-of-the-art English sentiment models are directly applied to the translated text. Several efforts were spent to explore the impact of translation on sentiment analysis in Arabic. Applying the ‘NRC’ model [54] to English translation of Arabic text achieved less accuracies than applying its “approximate” Arabic implementation to the original Arabic text [92]. Furthermore, applying the RNTN model [34] to English translation of Arabic text led to slight accuracy degradation compared to several baseline classifiers applied directly to Arabic text [93]. Despite the loss in accuracy due to inevitable translation errors, MT-based approaches are considered efficient alternatives to building sentiment models and resources in complex low-resource languages.

In summary, we can observe that most Arabic sentiment models depend on atomic and sparse bag-of-features, that result in fragile and non-robust systems. Also, previous efforts did not fully tackle the morphological complexity of the language, with one effort coming close to this scope by evaluating stem and lemma features, but without covering the full space of Arabic morphology [40]. We can also observe that most Arabic sentiment models were evaluated at the sentence and the document levels, despite the fact that sentiment is usually expressed in subtle manners and should be modeled at lower levels including words and phrases. Research on this direction is held back by the lack of lexical resources with fine-grained sentiment annotations that are required to train sentiment models at finer levels of granularity. At last, and to the best of our knowledge, state-of-the-art deep sentiment models were not explored for Arabic sentiment analysis. Therefore, an interesting task is to explore how they would perform if applied to Arabic text.



# Chapter 3

## Challenges and Objectives

In this chapter, we describe the challenges related to developing a meta-framework for opinion mining. We also describe the challenges associated with OMA, with a focus on relevance to recursive deep models. Finally, we conclude with the list of objectives that are accomplished in this dissertation.

### 3.1 Challenges Related to Opinion Mining Meta-Framework

Research in opinion mining has been driven by the need to improve the accuracy performance of existing models. In this dissertation, we conjecture that best accuracies can be achieved by models that come closest to the human reading. To the best of our knowledge, there have not been attempts to take a fundamental approach in matching the human process of extracting opinion from text. Therefore, a framework that builds its foundations from the field of psychology must be developed to determine aspects of the human reading process that need to be automated for opinion mining.

We expect that existing opinion models would exhibit one or more gaps with respect to the identified aspects of human reading. Therefore, approaches to automate the missing aspects and fuse them with the opinion model need to be designed carefully, taking into consideration the underlying learning structure of the model in hand. As a result, it is important to introduce an approach that qualitatively evaluates opinion models with respect to aspects of the human reading process, and that automates missing aspects and integrates them into these models to boost their performance

## 3.2 Challenges in OMA

### 3.2.1 Overview of Arabic Morphology

We start with a brief review of Arabic morphology, based on [94], to motivate the discussion to follow. Morphology, in any language, is concerned with the internal structure of the word. Arabic language is known for its rich morphology that interacts with both the orthography and the syntax of the language. Furthermore, Arabic morphology can be divided into two types: form-based and functional morphology [95].

1 **Form-based morphology:** Describes the form of the units that make up a word [94]. The central unit is the morpheme, which is the smallest meaningful unit in a language. Arabic language has different types of morphemes: concatenative and templatic morphemes.

- *Concatenative morphology:* Words are formed by sequentially concatenating three types of morphemes: 1) *stems* that are necessary for each word, 2) *affixes* that are directly-attached to stems, and 3) *clitics* that attach to stems after *affixes*. For example, the word وسينتصرون  $w+s+y-ntSr-wn$  ‘and they will win’ is formed by attaching the clitics  $+و$   $w+$  and  $+س$   $s+$ , and the affixes  $+ي$   $y+$  and  $+ون$   $+wn$  to the stem انتصر  $AntSr$ .
- *Templatic morphology:* Stems are formed using three types of templatic morphemes. 1) the *root*: a sequence of three, (sometimes four or five) consonants that describes an abstract meaning such as ح-ر-ن  $n-S-r$  ‘winning-related’, that is shared by all its derivations such as انتصر  $AntSr$  ‘he won’, منتصر  $mntSr$  ‘winner’, etc. 2) the *pattern*: an abstract template that is applied to roots and vocalisms, and is commonly represented by a string of numbers indicating the radicals’ positions and special symbols indicating the vocalisms’ positions (e.g., V). For example, the pattern ‘tV1V22V1V’ corresponds to تَفَعَّلَ, where the second radical is doubled. 3) the *vocalism*: specifies the short vowels (تشكيل, diacritics) in the pattern. For example, the stem مُتَصِّر  $muntaSir$  ‘winner’ is composed of the pattern ‘mV1tV2V3’ that is applied to the root ح-ر-ن and the vocalism  $uai$

2 **Functional morphology:** Describes the function of the units that make up a word, and how these units affect the word’s syntactic and semantic behaviour [94]. Arabic morphology has three types of functional operations:

- *Derivational*: Applied to create new words from other words, such that the core meaning of the original words is modified according to well-defined relations such as actor (اسم فاعل) and actee (اسم مفعول). Derivation typically involves a change in both pattern and part-of-speech. For example, the adjective نَاجِح *naAjiH* ‘successful’ with pattern ‘1A2i3’ derives from the verb نَجَحَ *najaHa* ‘he passed’ with pattern ‘1a2a3a’.
- *Inflectional*: Applied to create new words from other words by modifying the values of a set of inflectional features. The core meaning and the part-of-speech are not altered. Inflectional features include (person, gender, number, voice, mood, aspect, case and state). For example, the verb انتصرت *AntaSarat* ‘she won’, is an inflected form of verb انتصر *AntaSara* ‘he won’, with the “gender” feature being modified.
- *Cliticization*: Closely-related to inflectional morphology, but is optional and is only expressed using concatenative morphology. Clitics can come before the word (proclitics) such as the conjunction + و *w+* ‘and’, and can also come after the word (enclitics) such as the object pronoun + هم *+hm* ‘them’.

### 3.2.2 Arabic-specific Challenges

In this section, we describe the different challenges associated with sentiment analysis in Arabic. These challenges can be Arabic-specific, originating from the complexity of Arabic language. They can also be language-independent, relating to the structure of the recursive deep models.

#### Lexical Sparsity

As described in subsection 3.2.1, Arabic is a morphologically-rich language (MRL), in which multiple affixes and clitics can be attached in different ways to the words they modify. This leads to a higher degree of lexical sparsity than other non-MRLs such as English.

Using a large Arabic-English parallel corpus, it was observed that 1) the number of unique Arabic word forms is two times greater than that in English, and that 2) the number ‘whitespace-separated’ tokens in Arabic is 20% less than that in English [96]. The first observation is an implication of the complex morphological operations in Arabic, as modifying one or more inflectional/clitic features in a word produces a totally-new words. The second observation relates to the rich concatenative morphology in Arabic, where words usually contain many

morphemes, and hence are packed with significant amount of information. An example of this case is the word  $wa+sa+yu-nASir-wwna+hA$  وَسَيُنَاصِرُونَهَا, which corresponds to the multi-word phrase: ‘and they will support her’, and includes two proclitics, one enclitic, a prefix and a suffix, and the stem  $nASir$ . Moreover, the word-to-morpheme ratio in Arabic is two times greater than that in English, as observed in large corpora [97].

Arabic words are said to be an inflected form of the lemmas, which themselves are derived from the roots. Arabic has a relatively-small number of roots that derive a large number of lemmas, which inflect into a larger number of words. We combine in the sense of *word inflection*, both orthographic cliticization and inflectional morphology. In the context of sentiment analysis, we expect that inflectional variants sharing the same lemma or stem will maintain the same core meaning and sentiment, such as  $naASara$  نَاصَرَ ‘he supported’,  $yunaASir$  يَنَاصِرُ ‘he supports’,  $sayunaASir$  سَيُنَاصِرُ ‘he will support’, and so on. It is not necessary apparent that derivational variants sharing the same root will carry the same sentiment, however. Regardless, machine learning models that are trained using raw words will suffer from the high sparsity of the language, leading to poor generalization capabilities. For instance, a model that has learned the semantics of the word  $yantaSir$  يَتَنَصَّرُ ‘he wins’ cannot re-use this knowledge to understand new unseen words such as  $yantaSiraAn$  يَتَنَصَّرَانِ ‘they [dual] win’, although they share the same core sentiment.

## Lexical Ambiguity

In Arabic orthography, diacritics (عَلَامَاتُ التَّشْكِيلِ) are optional despite their key role at disambiguation by marking short vowels, nunation and gemination. This is the primary cause of Arabic’s notoriously high ambiguity rate, where SAMA morphological analyzer produces an average of 12.8 analyzes and 2.7 lemmas per word, out-of-context [98].

In many cases, words with identical forms can express different meanings and even different sentiments. For example, the undiacritized word  $E*b$  عذب can be interpreted as  $Ea* \sim aba$  عَذَّبَ ‘he tortured’, and also as  $Ea*obo$  عَذْبُ ‘sweet’. Another example that involves different interpretations of word tokenization complicated by dropped diacritics is the word  $b\$r$  بشر, which can be interpreted as  $ba\$ar$  بَشَرٌ ‘human’ (neutral),  $ba\$ \sim ara$  بَشَّرَ ‘he delivered good news’ (positive), and  $bi+\$ar \sim K$  بِشَرٍّ ‘with evil’ (negative).

Lexical ambiguity can also be a consequence of the idiosyncratic semantics of Arabic roots, whose meanings depend on the context in which they are used [94]. Therefore, words that are derived from the same roots can express different mean-

ings. For example, *مُصِيبَة* *muSiyyab* is derived from *ص-ي-ب* *S-y-b* ‘target-related’ can have a positive meaning as in ‘she is right’ or a negative meaning as in ‘a disaster’.

Based on this discussion, machine learning models trained using raw Arabic words will not be able to distinguish the different meanings the words may express. Although deep learning relies on word embedding to capture the distributional semantics of each word, these embeddings may have limited capabilities in face of extreme semantic variations of Arabic words that involve a change in sentiment polarity.

### **Inaccurate Word Embeddings**

NLP deep learning models use shared-task learning, where word embeddings are derived using a neural language model (NLM) and are then shared to perform different NLP tasks [99]. These embeddings are fine-tuned to the task in hand; sentiment classification in our case. According to [33], using pre-trained word embeddings achieved only marginal improvements compared to the case of using randomly-initialized vectors. This means that the fine-tuning process was able to produce representative word embeddings for the task in hand.

Although shared-task learning has proven successful in many NLP tasks such as POS tagging, Named Entity Recognition (NER) and Chunking [99], we suspect that these embeddings are not a good fit for sentiment analysis, and that better embeddings should be generated and used instead. This is mainly because NLMs use co-occurrence statistics in large corpora to derive the word embeddings that will consequently reflect latent aspects of the words, mostly related to their syntactics and semantics, but not their sentiment. Furthermore, NLMs are likely to derive similar embeddings to words with opposite sentiments, as long these words have similar usage in text. Examples include the pair of words ‘good/bad’, which usually appear in similar contexts; ‘the movie was good/bad’, ‘a good/bad weather’, etc.

Using word embeddings that are inaccurate in the sense of capturing the words’ sentiments will affect the outcome of compositionality, leading to sentence or document embeddings that are unable to capture the correct sentiment content of the text.

### **Sub-optimal Path of Composition**

Recursive deep models propagate the information up a binary parse tree in order to derive an overall embedding that captures the overall semantics and sentiment of the text. At each level of composition, the model combines two constituents that are selected based on the structure of a parse tree.

Several approaches exist to generate parse trees, among which stands the unsupervised “greedy” algorithm proposed by [33]. This algorithm discovers the

tree structure jointly while training the Recursive Auto Encoder (RAE) composition model. Although the resulting trees do not necessarily abide by the syntactic constraints of the language, this algorithm outperformed CKY-like beam search algorithms [100,101] in both accuracy and speed.

Given the rich concatenative morphology of Arabic, several levels of morphological tokenization exist, starting from raw words (no tokenization) to full tokenization where words are split into their basic morphemes (stems, affixes and clitics). Therefore, it is important to understand which tokenization level is better for composition. In other words, we need to determine whether morpheme-level interactions improve semantic composition, as opposed to raw word-level interactions.

On the other hand, given morphologically-tokenized text, the parse trees that are used to guide the composition should correctly link clitics and affixes to the stems they are attached to, which is not guaranteed by the greedy algorithm. As a result, it is important to use parse trees that combine, at each step, the constituents that are semantically and syntactically-related, to reflect the natural order in which words combine to express meaning in text.

### **Lack of Fine-grained Sentiment Resources**

Despite the fact that several corpora have been released with sentiment annotations in Arabic, there is still no corpus with fine-grained annotations at all levels of constituency. This is mainly due to the time, cost, and complex processing that are required. However, the availability of such resources will allow recursive deep models to incorporate sentiment information into the composition model, which has proved to yield significant improvements [34]. They will also open the way for more research on sentiment analysis in Arabic at different levels of text granularity (words, phrases, etc.).

## **3.3 Objectives**

Based on the previous discussion, we define the objectives that are accomplished in this dissertation. At the high-level, we aim to:

- 1 Develop a meta-framework that helps designing and/or extending sentiment models by matching the human reading process
- 2 Improve Opinion Inference models with focus on Arabic. This objectives encompasses the following sub-objectives:
  - Provide input features with accurate capture of sentiment
  - Improve the order of composition for sentiment analysis

- Overcome morphological complexity of Arabic and reduce their impact on sentiment analysis
- Mitigate the impact of lexical ambiguity on sentiment analysis
- Develop sentiment lexical resources to support recursive deep models at granular levels of text

Towards this objective, we intend to use recursive deep models as the learning algorithm. This will become the first deep learning-based solution for OMA.

# Chapter 4

## Human Reading for Opinion Mining

In this chapter, we take a fundamental approach in attempt to match the human reading process for opinion mining. We formulate opinion mining in a meta-framework that mimics the human reading process, and that can be used to either extend existing opinion mining models, or as a reference to develop new ones from scratch. This meta-framework is referred to as the Human Reading for Sentiment (HRS), and is considered one minute step towards bringing machines closer to the human-level performance. To the best of our knowledge, this is the first attempt to combine research from both opinion mining and cognitive psychology.

This chapter is organized as follows. In section 4.1, we provide a background on modeling the human reading in the field of cognitive psychology. In section 4.2, we motivate the need for HRS by identifying existing gaps in state-of-the-art opinion models with respect to human reading. Section 4.3 describes the application of HRS to extend existing state-of-the-art opinion models. Finally, experiments and results are presented in section 4.4.

### 4.1 Models of Human Reading

Despite significant advances in artificial intelligence (AI), humans are still better than machines at performing subjective tasks such as reading comprehension and information retrieval. Extensive studies in cognitive psychology have been carried out to model human reading comprehension. Some of these models try to describe the cognitive processes underlying the human reading process and how humans infer semantics. We are interested to use such models as a foundation for automatic interpretation of sentiment content in text.

Connectionist models represent a category of approaches that describe the cognitive processes behind mental phenomena in the form of cooperative and



competitive interactions among a larger number of simpler units. An example of such approaches is the Incremental Construction of Associative Network (ICAN) model [102], which regards reading as a process of sequential perceptions over time. Within this process, the human mind builds mental images and inferences that keep reinforced, updated or discarded, and when readers reaches the end of text, they use the resulting image to summarize, classify or infer meaning. In another connectionist model, human readers comprehend a text and access its semantics, through a direct mapping from orthography to semantics by way of visual access, or through a mediated mapping through phonology. Based on the “triangle framework” [103], reading comprehension was regarded as a joint cooperative division of labor that depends on factors such as homophony and visual similarity [104]. Connectionist models have also been used to model different reading-related tasks including reading aloud and word recognition [105–107].

The Direct and Inferential Mediation (DIME) model of reading comprehension is another model that attempted to study the human reading process. DIME hypothesizes relations among five predictors of comprehension: word visual reading, vocabulary, background knowledge, inference and comprehension strategy, and describes the direct and indirect effects of these components on comprehension [108]. The landscape model (LM) [109] considers reading a balancing act between the readers limited working memory (WM) and the need for coherence. Attention is influenced by the availability of WM and relevant background knowledge. Limitations in WM capacity, and the necessity of simultaneous activation to infer relations, make the readers attention allocation of great importance for reading comprehension. Finally, the model proposed by [10] assumes that humans perform two types of cognitive processes during reading. At the lower-level, they use their syntactic knowledge to come up with textual representations of concepts or ideas, and at the higher-level they use their background knowledge and working memory to infer the overall meaning of the text.

Automating the human reading process is equivalent to machine reading (MR), which refers to the task of “understanding text” by automatically forming a coherent set of beliefs based on a textual corpus and a background theory [110]. Although the humans’ ability to grasp complicated nuances from text greatly surpasses that of a machine, MR still has several strengths; it is fast and can leverage statistics from large-scale corpora [110]. MR includes sub-problems such as information extraction, question-answering, text summarization and sentiment analysis. Different tools have been developed for MR such as KnowItAll [111], TextRunner [112] and Kylin [113] for open-domain information extraction, Mulder [114] for web-scale question answering and HOLMES [115] for inference.

It can be observed that most of the described reading models share similar concepts, mainly the importance of background information, working memory, and the retaining of relevant notions. Our interest was to use such models as a foundation for automatic interpretation of sentiment content in text. We chose to automate the reading model proposed by [10] because it also regards the reading

process from a linguistic point-of-view, and describes how humans use their grammatical and syntactic knowledge towards inference, which is often downplayed by other models. Furthermore, this model provides a clear description of the human reading using a sequence of steps (processes), as illustrated in Figure 4.1, which makes it more straightforward to automate.

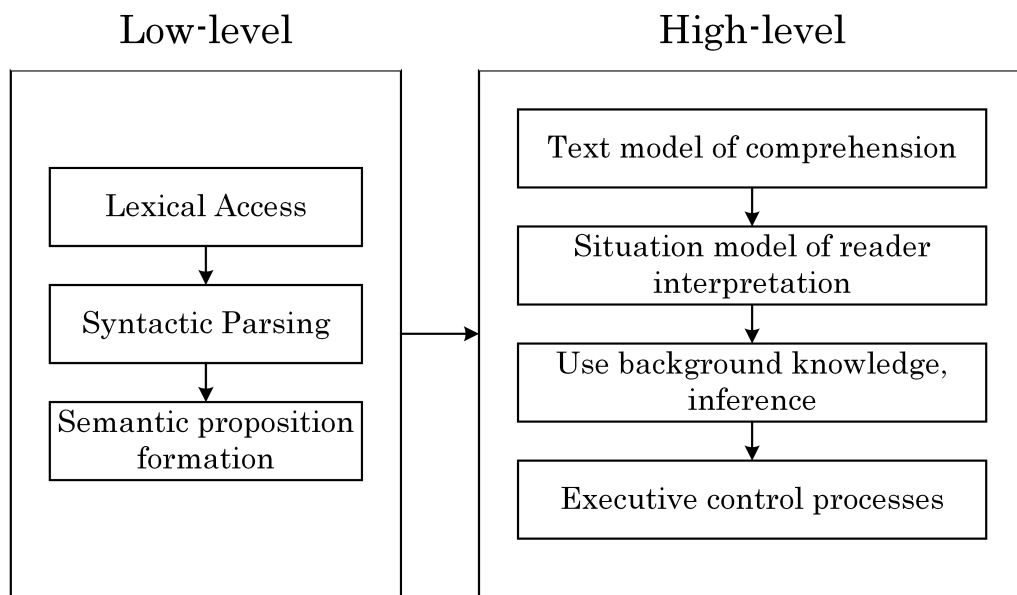


Figure 4.1: The human reading process.

At the low-level processes, readers process the document at granular levels. They first perform lexical parsing, which is the process of converting a sequence of characters into a sequence of tokens. The next step involves syntactic parsing based on the grammatical rules by which sentences are constructed. Finally, readers develop a semantic proposition which bases itself on lexical and syntactic parsing to form sub-sentence structures called clauses, each representing a certain idea or concept.

At the high-level processes, readers build an understanding of the individual clauses and the document as a whole. First, readers use the purpose of reading with their background knowledge to decide which clauses can represent the document. Clauses that do not fit the purpose of reading are discarded, and the remaining clauses form the “*text model of comprehension*” which is the parts of the text that are of interest to the readers. Based on the readers’ background knowledge and preconceived notions on the topic, the “*text model of comprehension*” is transformed to the “*situation model of reader interpretation*”. This is equivalent to assigning a sentiment label to each clause in the “*text model of comprehension*”. As a result, notions correspond to ideas represented by the combination of textual clause representation and their equivalent sentiment. Once

readers have inferred notions from assigned sentiment labels to each clause, they can consequently infer the sentiment orientation of the document as a whole. Inference takes several steps in the human cognition, and is an important factor that reflects how humans reach to conclusions through reading [116]. To infer the overall meaning of a document, readers must have certain facts or evidence in their memory when reaching the end of the text. Several models were proposed to describe the impact of the working memory on inference. According to [117], the reading task requires maintaining significant amount of memory much more than the capacity of the working memory. Once the working memory is full, it becomes difficult to understand the complex relations between notions. Therefore, humans store most of what they read in their long-term memory and link them together through retrieval structures. Consequently, the working memory holds only a few concepts that serve as cues to retrieve related information via retrieval structures. As a result, the retaining of facts is part of the working memory (WM) that combines the most recent information stored in the short-term memory (temporary storage) with information manipulation during reading comprehension to make sense out of the text [118]. The working memory directly affects reading comprehension, where humans with low working memory capacity face difficulties in comprehension [119]. For instance, while a child with low WM is working hard to decode written words, he may lose track of the overall purpose of the text.

We use the above-mentioned model as the foundation to develop the HRS meta-framework for automating the human reading process for sentiment inference. The conforming framework must accurately capture the low-level and high-level cognitive processes illustrated in Figure 4.1. In this dissertation, we use the HRS to extend and improve existing sentiment analysis models following a two-step approach. We focus on document-level sentiment analysis in order to highlight the impact of working memory on inference. First, a qualitative analysis of the model under consideration is performed against the human reading steps to identify potential gaps (discussed later subsection 4.2). Then, approaches to address the identified gaps are developed and integrated into the model. These approaches vary depending on the underlying learning scheme. For instance, and at the high-level, addressing gaps in feature engineering-based models can be done by proposing new features or by modifying the classifier, whereas addressing these gaps in deep learning-based models can be done by modifying the network’s architecture or its raw input data.

## 4.2 Research gaps with respect to HRS

In this section, we go deeper into the state-of-the-art models on document-level sentiment analysis, and provide a qualitative analysis to highlight the human reading-specific gaps, with the aim of highlighting gaps that need to be addressed

by HRS. The objective is to evaluate how well each method captures each of the different human reading steps mentioned in section 4.1, namely (1) syntactic parsing (the use of the language grammar), (2) semantic proposition (the use of constructs that match concepts or ideas), (3) text model of comprehension (the retaining of relevant ideas), (4) situation model of interpretation (the use of sentiment to label ideas) and (5) inference and proper modeling of memory.

Results of the qualitative analysis are summarized in Table 4.1, where columns of the table reflect the different aspects of the human reading process, and rows list the most relevant models proposed for document-level sentiment analysis. For example, [120] proposed to predict document-level sentiment while jointly selecting the key sentences in every document. This method clearly captures and optimizes the “*text model of comprehension*”, but misses to model the remaining aspects. The method proposed by [58] assumes that documents must be represented by selected subjective clauses, which is consistent with the “*semantic proposition formation*” and the “*text model of comprehension*”. However, the model does not incorporate sentiment in these clauses, hence fails to capture the “*situation model of interpretation*”. Another example is the method proposed by [66], which is based on deep learning. This model proposes a document embedding that is inspired by word embedding techniques, with the purpose of creating a document vector representation that captures the semantics and context of all words in that document. This model incorporates information from all words and sentences, with equal considerations, into the document vector, hence the method does not meet the “*text model of comprehension*” aspect, which assumes that some parts of the document are more important than others. Additionally, the resulting document representation reflects the syntactics and semantics of its composite words, not their sentiments, hence it does not capture the “*situation model of interpretation*” aspect. Other methods in Table 4.1 exhibit different gaps in comparison with the targeted HRS. These results indeed confirm that previous methods that were developed to improve document-level sentiment analysis handle some aspects of the human reading process, but not all. The proposed HRS provides a unifying meta-framework to address the human-reading gaps with any existing method.

The design and structure of HRS depend on the underlying learning scheme. The details of adjusting different sentiment analysis methods to HRS are presented in the next section. Furthermore, the effectiveness of HRS is quantitatively evaluated in Section 4.4 on two selected state-of-the-art methods, namely FRN [12] and GRNN [13]. FRN was selected as a representative of machine learning models that are based on feature engineering, and GRNN [13] was selected as a recent state-of-the-art technique that represents the latest advances in deep learning, and that achieved highest performances on different benchmark datasets.

	<b>Syntactic Parsing</b>	<b>Semantic proposition formation</b>	<b>Text model of comprehension</b>	<b>Situation model of interpretation</b>	<b>Inference and Memory</b>
[120]	N/A	N/A (classify based on <i>uni</i> -grams)	Optimized solution to determine important sentences in a review	N/A	Classify reviews using <i>uni</i> -grams in identified important sentences
[12]	Use POS tag n-grams and word-POS tag n-grams	Represented by n-grams	FRN keeps only relevant features after reduction	N/A (capture semantics, not sentiment)	SVM trained with relevant n-grams (equally-important)
[58]	Perform syntactic parsing to represent reviews' sentences	Identify substructures (clauses) from parse trees	Assume reviews must be represented by selected subjective clauses	N/A	SVM trained with subjective clauses represented by trees kernels (equally-weighted)
[66]	Word-level syntactic properties captured via embedding	N/A (derive document vector using word vectors)	N/A (used all words to derive the document vector)	N/A (word embedding captures semantics, not sentiment)	Softmax trained with derived document vectors
[65]	Word-level syntactic properties captured via embedding	Derive document vector using 1- and 2-grams	N/A (used all words to derive document vector with stacked denoising AE)	N/A	SVM trained with document vectors derived using SDA
[121]	Word-level syntactic properties captured via embedding	Use CNN with multiple filters to capture n-grams semantic	N/A	N/A	Softmax trained with document vectors (avg. sentence vectors) and author/product features
[13]	Word-level syntactic properties captured via embedding	Use CNN with multiple filters to capture n-grams semantic	Weights optimized in CNN to select which n-grams to use for modeling	N/A	GRNN derives document embeddings and models the impact of each sentence

Table 4.1: Qualitative analysis of document-level sentiment analysis models against aspects of human reading.

## 4.3 Applications of the HRS Meta-Framework

In this section, we describe the proposed HRS meta-framework to automate human reading with primary focus on understanding sentiment instead of the full semantics of the text. To demonstrate the effectiveness of HRS, we apply it to two state-of-the-art methods. The first method, FRN [12], is chosen to represent methods that take a two-step approach, the first step being feature engineering and the second step being classification or inference. On the other hand, GRNN [13] is chosen to represent deep learning approaches that derive semantic features from raw data without going through feature engineering. It is worth mentioning that frameworks resulting from the HRS meta-framework must be consistent with the learning method being used. Consequently, HRS steps for feature engineering-based methods would be different than HRS steps for deep learning-based methods. Next, we describe the proposed application of HRS to FRN and GRNN sentiment analysis methods.

### 4.3.1 HRS with Feature Engineering: FRN as Case Study

[12] explored a wide range of surface, stylistic syntactic and semantic features, and proposed the Feature Relation Network (FRN) algorithm to retain only the relevant and non-redundant ones. FRN ranks the features using semantic information about the subjectivity of each feature. It also derives a score that represents the discriminating power with respect to the different classes. Then, every feature is assigned a weight that is equal to the sum of both semantic and discriminating scores. These weights are then used by the subsumption and parallel relations to remove redundant or irrelevant features that do not convey any extra information to the classification process. The resulting feature subset is then used to train an SVM classification model. Further details of the approach are available in [12]

First, we perform a macro-level analysis to identify gaps in FRN with respect to the human reading process. FRN uses a wide range of features and captures well low-level processes through the NLP task of extracting shallow text features. To automate “*lexical and syntactic parsing*”, FRN uses a syntactic base created through tokenization and POS tagging. However, it partially captures the “*semantic proposition formation*” by extracting  $n$ -grams instead of phrases. It also fails at modeling the high-level processes of “*semantic proposition formation*”, “*situation model of interpretation*” and “*working memory*”. In other words, FRN captures most of the low-level processes, but misses the high-level processes.

The second step in the method is to address the identified HRS-related gaps in FRN. Figure 4.2 illustrates the framework that is proposed to incorporate the missing HRS aspects, highlighted in gray. Instead of using  $n$ -grams to model the “*semantic proposition formation*”, we perform phrase extraction by splitting

sentences according to the presence of coordinate and subordinate conjunctions (but, for, etc.) and punctuation [122]. To address gaps in the high-level processes, we propose to extract the “notions” features. This stage of semantic analysis requires the development of: (1) a domain lexicon that contains frequent keywords relevant to the domain, to model the “*text model of comprehension*”, (2) a background knowledge database of notions; assigns sentiment scores to selected phrases, to model the “*situation model of interpretation*”, and (3) a weighting scheme that assigns a memory score to each notion based on its location in the document, to emulate the impact of the “*working memory*”. At last, inference is achieved through a classification model that determines the sentiment polarity of the document. This classifier is trained using the proposed notions features, in addition to the syntactic and semantic features obtained from FRN.

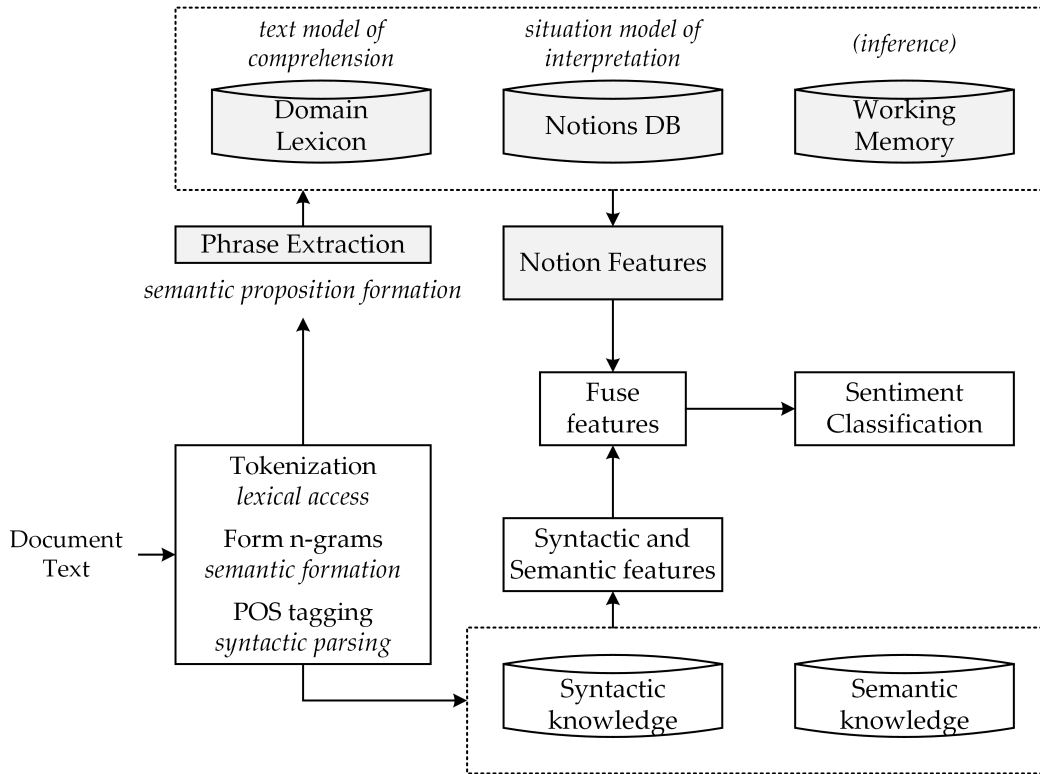


Figure 4.2: Application of the HRS meta-framework to the FRN method.

Below, we describe the details of the resulting HRS framework that is proposed to address the human reading gaps in FRN. In particular, we describe the feature set that is reduced by FRN [12] to model the low-level processes in the human reading. We also describe the proposed approach to extract the notions features to model the high-level aspects of the human reading process.

## Syntactic and Semantic Features (for low-level processing)

The FRN method [12] uses a rich set of syntactic and semantic  $n$ -gram features. We will refer to these features as the ‘FRN’ features. The following is a brief description of the different features.

- $n$ -words and  $n$ -characters: Sequences of characters and words of different lengths  $n$  ranging from 1 to 3.
- $n$ -POS tags: Sequences of POS tags of different lengths  $n$  from 1 to 3.
- $n$ -POSwords: Sequences of words and POS tags of different lengths  $n$  from 1 to 3.
- $n$ -legomena: Consist of normal  $n$ -word features, with the difference that words occurring only once are replaced with the word “HAPAX”, and words occurring only twice in the corpus are replaced with the word “DIS”.
- $n$ -semantic: These are  $n$ -words where words that belong to some synonym group in WordNet are replaced by their group label. The label becomes a unique ID identifying the specific synonym group.
- Information Extraction Patterns: A set of syntactic templates that indicate subjective content (e.g., passive-verb: “was destroyed”) [123].

This feature set is huge, reaching millions of features for moderate-size corpora. Hence, it is necessary to perform feature reduction in order to keep the relevant ones for later use in the HRS model. Many feature selection algorithms have been proposed for sentiment analysis and text categorization including Entropy-weighted genetic algorithm (EWGA) [18], feature relation networks (FRN) [12], and feature ranking using different measures such as proportional difference of SentiWordNet scores [124], information gain,  $\chi^2$ , document frequency [125], and standard deviation [126]. We decided to use the FRN reduction approach, which already achieved high performance when used to reduce the FRN features set.

## Notion Features and Background Knowledge

These features are proposed to capture the following human reading aspects: the “*semantic proposition formation*”, the “*text model of comprehension*” and the “*situation model of interpretation*”. In general, the human’s preconceived notions enable a person to decide whether a text has positive, negative or neutral sentiment. We define the equivalent of a human’s preconceived notions on a topic as the combination of the pairs of: {topic textual characteristic, sentiment score according to the human subject}. For example, {*low battery consumption*, (+) score} and {*low resolution*, (-) score} represent positive and negative notions in



someone’s mind, respectively. To define the notions’ textual characteristics, we assume that the smallest text unit that contains a notion is a phrase that is composed of adjectives, adverbs, nouns and verbs. Phrases are identified in sentences based on the presence of known coordinate and subordinate conjunctions (and, but, for, etc.) as well as punctuation including commas and periods.

However, not all phrases can be considered as notions in a particular domain, as some could present ideas that are general or irrelevant in building the set of domain-specific notions. A domain lexicon is developed to evaluate the relevance of a phrase to become a notion. The domain lexicon contains common nouns and verbs that appear frequently in a particular domain. It is used to check whether each phrase contains any domain-related nouns or verbs, which would have an impact on the overall sentiment of the document. Consequently, the notions database is formed by going through the documents and storing phrases that contain at least one domain word, be it noun or verb.

Each phrase (the notion’s textual representation) needs to be associated with a sentiment score that indicates the degree of negativity to positivity expressed by the phrase. This score is automatically obtained for each phrase by calculating the difference in point-wise mutual information between that phrase and the sentiment classes [54], as illustrated in Equation (4.1).

$$\begin{aligned} \text{Sentiment score of phrase } p &= PMI_{p,pos} - PMI_{p,neg} \\ &= \log_2 \left( \frac{freq(p, pos) \times freq(neg)}{freq(p, neg) \times freq(pos)} \right) \end{aligned} \quad (4.1)$$

where  $freq(p, pos)$  is the number of times phrase  $p$  occurs in positive documents,  $freq(p, neg)$  is the number of times it occurs in negative documents,  $freq(pos)$  and  $freq(neg)$  are the count of positive and negative documents in the corpus, respectively. The sentiment scores are then normalized to fall in the range between  $[+1, -1]$ . The resulting background knowledge database consists of phrase notions related to a particular domain. When a new phrase is encountered, it is compared against the background knowledge database to match with the textual characteristics of the notions it contains. The matched notions then make up the document’s notion features which are used to infer the document-level sentiment.

It is worth mentioning that the background knowledge database is in a way similar to sentiment lexicons with the following distinctions. (1) It consists of phrases while most current sentiment lexicons consist of words or word n-grams. (2) Its entries pertain to a particular domain rather than being generic, which allows it to model the humans’ notions on a particular topic. (3) It is automatically generated and does not involve manual annotation.

### **Notions Synonyms (for improved generalization)**

When classifying new text, the proposed algorithm tries to recognize notions that are previously stored in the background knowledge database. For training, the matches help in grouping similar notions together in the notions database. For classification, the matches help assess the sentiment polarity by measuring similarity to a previously stored notion. However, the algorithm would not detect matches between variations of the same notion carrying the same meaning. In language arts, different people may express the same idea through different words, hence a limited set of notions expressions without synonymous notions would limit the matching process. For this reason, it is important to store the notions in a way that allows generalization of notions, supporting proper semantic matching of notions with either exact word matches or synonymous semantically-related words.

WordNet [22] has been widely used in sentiment analysis systems as a rich source of semantic relations between words, providing generalization capabilities. For instance, words were replaced by the set of their hypernyms in WordNet in the task of subjective expression identification [127]. *Uni*-grams were replaced by their equivalent synonym sets (synset) labels to create a set of semantic features [128]. Synsets were also used to create semantic categories by clustering words based on the number of common items in their synsets [129]. Every new word is added to the cluster with the highest percentage of common synonyms.

We propose to use WordNet synsets to create a database of notions' synonyms. The textual characteristics of existing notions are replaced by synonymous generalized text, which are combined with the sentiment scores from the original notions to create the notions' synonyms (SYN notions). For instance, notions with the following textual characteristics: "*passion movie*" and "*love movie*" are synonymous since they convey the same meaning. According to WordNet, the words *passion* and *love* belong to the same synset whose label is 40. Therefore, when creating the SYN notions, each of these words is replaced by their synset label in WordNet, i.e., 40. As a result, the textual characteristics of both notions are now represented by "*SYN40 movie*".

### **Human Working Memory (for improved inference)**

The goal of modeling the working memory is to bring the automation process closer to the way a human reader processes, analyzes and draws conclusions from a document. As readers progress through text, they are more likely to remember later sentences which are stored in short-term memory [118]. Consequently, we propose to assign each notion in a document a weight that models the impact of the human working memory. The weighting scheme assigns weights depending on the notion's position within the document; weights are low for notions that appear at the beginning, and become higher for notions that appear at later

parts. This weighting scheme is illustrated in Equation (4.2).

$$\text{working memory weight of notion } p = \left( \frac{\sum_{i=1}^{n_p} loc_i}{n} \right) \quad (4.2)$$

where  $n_p$  is the frequency of notion  $p$  in the document,  $loc_i$  is location of the  $i^{th}$  occurrence of notion  $p$  in the document, and  $n$  is the count of all notions in the document. Then, each notion is represented by the product of its sentiment score and its memory weight to reflect the fact that a reader’s sentiment interpretation would be affected by what they read at the end more than what they read at the beginning.

Several previous works evaluated the importance of later sentences based on the hypothesis that authors tend to summarize their ideas and opinions at the end of the text [15,130]. It also confirms the necessity of the working memory for sentiment comprehension. According to [130], by reading only the last sentence of a review, human readers were able to predict the reviews’ sentiment polarity with performance that is comparable to the case when they read the whole review. Excluding objective sentences prior to sentiment analysis was proposed by [15]. Among the top-ranked objectivity filters was the one that keeps the last  $N$  sentences assuming their importance in summarizing the text sentiment.

The proposed working memory weights differ from previous work on weighting ideas in the following aspects. First, it is applied at the notion or phrase-level rather than the whole sentence-level. Second, it assigns gradual weights instead of a crisp decision of whether or not to include certain notions into the sentiment analysis model, hence it takes into consideration all notions in the document, with varying weights. This somehow remedies the issue that arises when analyzing documents with no concluding statements. At last, it is worth mentioning that the proposed working memory feature is expected to work best with a particular genre of text that contains a conclusion in the last sentences. Consumer reviews constitute a major portion of this genre.

### 4.3.2 HRS with Deep Learning: GRNN as Case Study

#### HRS Applicability to Deep Learning

Deep Learning (DL) covers a wide range of models with different architectures and different capabilities of capturing psychological aspects of the human reading, including the low-level processes (“*lexical access*”, “*syntactic parsing*” and “*semantic proposition formation*”) and the high-level processes (“*text model of comprehension*”, “*situation model of interpretation*” and “*inference*” including impact of working memory).

DL models capture the low-level processes as they perform “*lexical access*” by being applied at the word-level, and recently at the character-level. They indirectly incorporate grammatical knowledge via word embeddings, which capture distributional syntactic and semantic information of the words and are also used to train DL models. The “*semantic composition formation*” is also captured in recursive DL models that use composition functions to derive the meanings of phrases (concepts or ideas), which are later used to derive the sentiment of the text.

Regarding the high-level processes, most DL models do not capture the “*text model of comprehension*”, as they do not select phrases ideas, based on their relevance to the purpose of reading. The “*situation model of interpretation*” aspect is captured by many DL models that capture sentiment in the word and phrase representations, which in turn become equivalent to notions (or a combination of a textual representation and a sentiment indicator). The RNTN model we used in this thesis is an example of DL models that capture this aspect. Finally, DL models model the “*Inference*” aspect by modeling semantic compositionality, where words and phrases are combined together, recursively or sequentially, in order to derive the overall meaning and sentiment of the text. Compositionality is a more realistic approach, that comes closer to the way humans perform inference than training classical machine learning models with n-gram features and different choices of feature engineering. Furthermore, the “*working memory*” can take part of the inference process as recent advances in DL, namely the Long Short-Term Memory (LSTM), allow to capture long-distance dependences between phrases and decide when should a phrase be remembered or forgotten while inferring the overall meaning or sentiment. Recent results in English have shown that LSTM achieve significant improvements compared to models that do not model the memory aspect. Recurrent models also capture the impact of memory, as they try to incrementally, or sequentially, develop the meaning of the text, where at each point of reading, they combine the current word with the representation of all its preceding words that were read.

## **HRS Applicability to GRNN**

The GRNN [13] is a deep learning model that extracts sentiment from raw data using dense, continuous and low-dimensional input word vectors, referred to as word embeddings, that capture latent syntactic and semantic aspects of the words [64]. The GRNN has a hierarchical architecture of three stages. First, Convolutional Neural Networks (CNN) [131] are used to perform semantic composition by transforming the sequence of the sentence’s word vectors into a single vector representation that embeds the overall semantics of the sentence. Then, recurrent Long Short-Term Memory (LSTM) is used to derive a vector representation of the overall document using its sentence vectors that are obtained in the first stage. At last, the resulting document representation is used to train a

logistic regression “Softmax” classification layer to predict the sentiment polarity of the document. To extend and improve the GRNN according to the HRS meta-framework, we follow the same strategy of identifying the human reading-specific gaps in GRNN, and then proposing approaches to address these gaps.

According to the qualitative analysis presented in Table 4.1, the GRNN captures several aspects of the human reading process. For instance, the word embedding is responsible for lexical access and syntactic parsing. The CNN model uses three convolution filters that span over windows of 1, 2 and 3 words to encode the semantics of  $n$ -grams. This is equivalent to the “*semantic proposition formation*” step, with the phrase length being limited to *tri*-grams. The recurrent LSTM model that is used in the second stage of the hierarchy learns long and short-term events by including a “forget” gate acting as a switch that erases or keeps the dependency between the current sentence and past ones. Hence, the network learns a model that acts similar to the working memory, where human readers develop their understanding of text by continuously retrieving or discarding previous information, based on their contribution to the text semantics. The main gap that can be observed in GRNN is the lack of modeling the “*situation model of interpretation*”, where the intermediate phrase and sentence representations do not capture sentiment information. Another limitation is the inability to perform “*semantic proposition formation*” and capture ideas represented by phrases longer than three words, due to the restricted size of the CNN filters. We focus on addressing the “*situation model of interpretation*” gap, as the other issue does not represent a major obstacle towards incorporating HRS with GRNN.

Figure 4.3 illustrates the GRNN architecture with and without HRS. We propose to address the “*situation model of interpretation*” gap by developing a word-level sentiment embedding block to derive word vectors that capture the sentiment polarity of each word. After obtaining the word “sentiment” vectors, each word is then represented by concatenating both its traditional and sentiment embedding vector. The resulting word vector representations would then capture syntactic, semantic and sentiment aspects of each word. Then, the CNN maps this information to higher-level constituents, or phrases, to derive the sentence-level vector representations. These phrases, along with their embedded sentiments, are equivalent to the notions limited to three words in length.

To perform word sentiment embedding, we propose a neural network architecture that is trained with the objective of (1) predicting the sentiment polarity of the individual words, each represented by a  $d$ -dimensional vector, and (2) fine-tuning the words’ vectors to capture their sentiment polarities.

The input layer contains  $d$  neurons, where  $d$  is the dimensionality of the word vector, and is a hyper-parameter that is either user-defined or tuned during model development. The output layer contains three neurons, each with a “softmax” activation function, to predict the sentiment polarity of the word, whether it is positive, negative or neutral. Typically, the output layer generates, for each word, a sentiment distribution  $\mathbf{y} \in \mathbb{R}^3$ , which indicates the probability of each

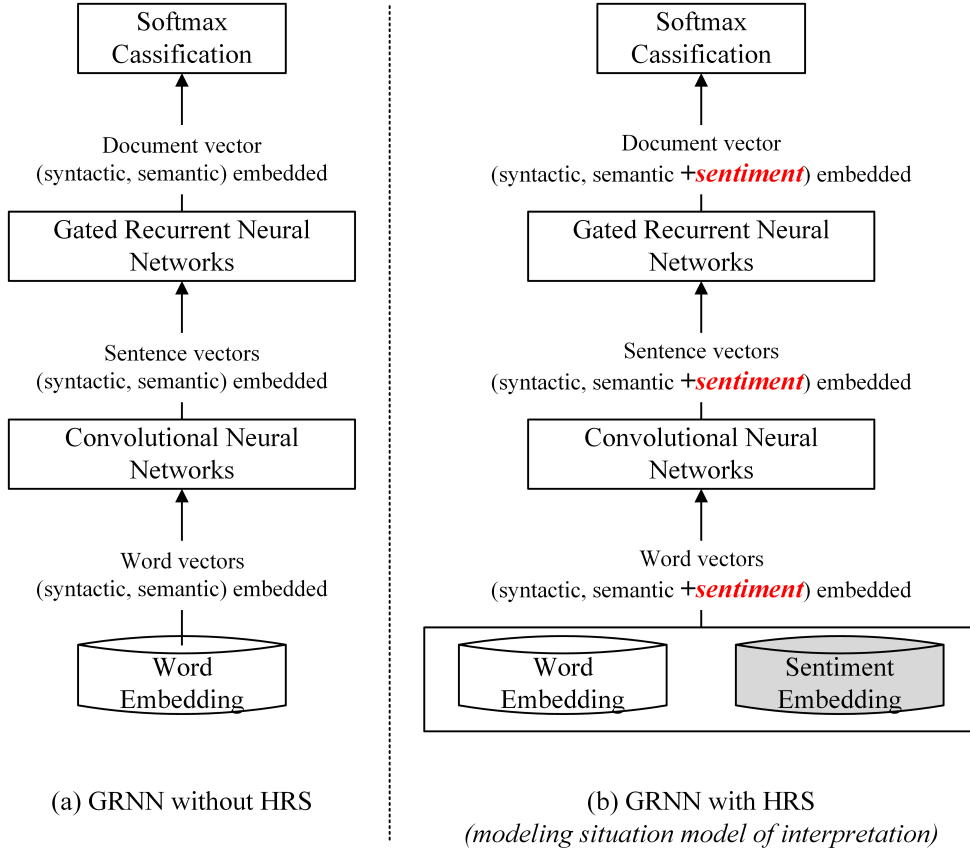


Figure 4.3: The architecture of GRNN with and without HRS. (a) shows the original GRNN framework. (b) shows the modified GRNN framework with the modeling of the “*situation model of interpretation*”

sentiment given that word. The network’s parameters are updated to minimize the cost function  $J$  corresponds to the error between the words target and predicted sentiment distributions, as shown in Equation (4.3).

$$J = \min \sum_{i=1}^V (\mathbf{t}_i - \mathbf{y}_i)^2 \quad (4.3)$$

where  $V$  is the size of our vocabulary (i.e., the words for which we want to learn sentiment embeddings),  $\mathbf{t}_i$  and  $\mathbf{y}_i$  are the target (gold) and predicted sentiment distributions of the  $i^{th}$  word, respectively. For our experiments, we trained the sentiment embedding network using words extracted from the SentiWordNet [21]; a sentiment lexicon that assigns to each word three sentiment scores indicating its degree of positivity, negativity and neutrality.

In addition to learning the network’s parameters, we also fine-tune the  $d$ -

dimensional input word vectors so they become representative of the sentiment of these words. Fine-tuning the word vectors is equivalent to performing an additional step in the back-propagation algorithm to update the activation values of the neurons in the input layer. The desired output of this sentiment embedding block is the set of fine-tuned word vectors that can be represented using a matrix, or a look-up table,  $L \in \mathbb{R}^{V \times d}$ , where  $V$  is the size of the vocabulary and  $d$  is the size of the embedding vectors.  $L$  can also be regarded as a fully-connected network of  $V$  input words, and  $d$  output neurons. The output of this network is equivalent to the input of the sentiment embedding network. Given an input word  $w_i$ , we get its equivalent vector using the lookup operation shown in Equation (4.4).

$$\mathbf{x}_i = \mathbf{e}_i \cdot L \quad (4.4)$$

where  $\mathbf{x}_i \in \mathbb{R}^d$  is the vector of the  $i^{th}$  word,  $\mathbf{e}_i \in \mathbb{R}^V$  is the one-hot encoding vector whose elements are all zeros, except for the  $i^{th}$  element that activates only the  $i^{th}$  word in the vocabulary. After each complete round of feed-forward and back-propagation, the look-up table is updated with the new word vector.

## 4.4 Experiments and Results

In this section, we apply the HRS meta-framework to two state-of-the-art methods in document-level sentiment analysis. We show that applying HRS to each method yields improvement even when performance is already high without HRS. The analysis focuses on two approaches, FRN and GRNN, representing machine learning approaches with focus on feature engineering and deep learning, respectively. The datasets used with each method are consistent with the choices made by the respective papers, enabling performance comparison for HRS versus these previous methods as standalone. Subsection 4.4.1 describes the datasets used in the experiments and the evaluation methods and measures. Subsections 4.4.2 and 4.4.3 illustrate the impact of incorporating HRS with FRN and GRNN models, respectively.

### 4.4.1 Datasets and Evaluation

Two datasets are used for the quantitative analysis; IMDB movie reviews and YELP restaurant reviews. The IMDB dataset is used by FRN and contains 2,000 reviews equally-distributed across positive and negative sentiment classes [14]. The YELP dataset contains 10,000 restaurant reviews selected from the YELP dataset <sup>1</sup>, which was also the source data for GRNN. Reviews in this dataset come with human-based ratings on a scale of 1 to 5. Table 4.2 highlights the characteristics of these datasets.

---

<sup>1</sup>from the 2013 YELP dataset challenge, [www.yelp.com/dataset\\_challenge](http://www.yelp.com/dataset_challenge)

Corpus	Size & Domain	Class Distribution	Splits		
			Train	Dev	Test
IMDB	2K movie reviews	50% - 50% (positive vs. negative)	70%	10%	20%
YELP	10K restaurant reviews	9% - 9% - 14% - 33% - 35% (very neg, neg, neut, pos, very pos)	80%	10%	10%

Table 4.2: Characteristics of datasets used in experiments.

For evaluation, the IMDB dataset was randomly divided into a train set (70%), a development set (10%), and a test set (20%). As for YELP, the splits had the following sizes (80%-10%-10%) for train, development and test, respectively. To tune the classifier’s parameters, the train and development sets were used. Then, for final model evaluation, the train and test sets are used with the parameters that achieved highest performance during tuning. Results are reported in terms of classification accuracy and F1-score as applied on the unseen test splits.

Software and tools that were used to conduct the required experiments are mentioned in Appendix A.

#### 4.4.2 Evaluating FRN with HRS

As previously discussed, FRN does not model each of the “*semantic proposition formation*”, the “*situation model of interpretation*” and the “*working memory*”. To model these aspects with FRN, the HRS steps consist of (1) extracting phrases, (2) retaining domain-relevant phrases using domain lexicon, and (3) assigning each phrase a sentiment score and a working memory weight. Notion features are then fused with the set of syntactic and semantic features to train a sentiment classification model. In the experiments, nonlinear SVM with radial basis function kernel is used for classification.

#### Evaluating Text Model of Comprehension

To evaluate the text model of comprehension, we consider different parts of the domain lexicon, and study the effect on overall performance. We simultaneously use this experiment to get the tuning parameter for the determination of the size of the domain lexicon. This lexicon contains frequent nouns and verbs that pertain to the domain being discussed. To determine the frequency thresholds that should be used to develop the domain lexicon, we performed a set of tuning experiments by training SVM using domain-relevant phrases with different possible thresholds. Thresholds are determined as follows: after removing stop-



words, nouns (or verbs) are ranked based on their frequency, and the frequency threshold is identified as the one that allows retaining the most frequent nouns (or verbs) that constitute  $X\%$  of the total count. In these tuning experiments, the choices of  $X$  thresholds are set at 100%, 90%, 80%, 70% and 60%. Also, both the domain lexicon entries and the phrases are extracted from the training set, to avoid overfitting. Results in Figure 4.4 show that using a domain lexicon whose entries constitute 80% of the total count of nouns and verbs in the corpus yields the highest performance as measured in both accuracy and F1-score. This observation confirms the concept of the “*text model of comprehension*” in the human reading framework, which states that only domain-relevant phrases should be used. It can be observed that using  $X = 100\%$  does not yield the best performance given that many phrases can be domain-irrelevant. Also, reducing  $X$  down to 60% decreases performance as many domain-relevant phrases are excluded from the model. These results are consistent with the classic bias-variance tradeoffs in prediction problems, with 80% achieving the best tradeoff between using enough of the data to improve performance, but not too much to avoid over-fitting.

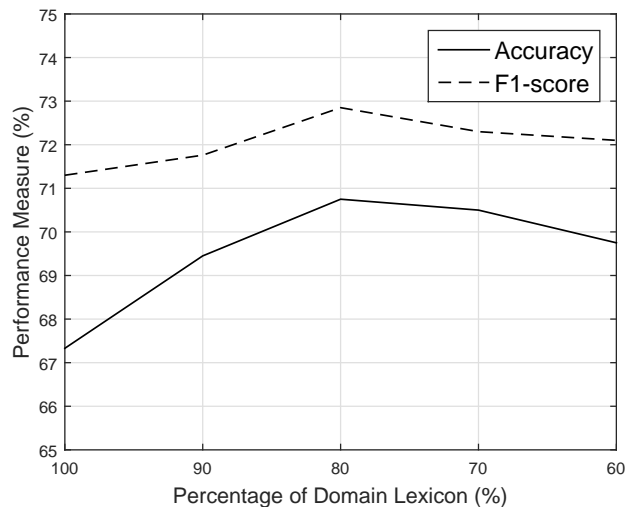


Figure 4.4: Evaluating different sizes of the domain lexicon.

### Evaluation of Synonyms

For the purpose of evaluating the impact of synonyms, two SVM classifiers are trained separately with either phrases or SYN-phrases that are extracted from the training set. This experiment is repeated for different sizes of the domain lexicon, similar to the previous experiments. Results in Figure 4.5 show that using SYN-phrases yields better results, in both Accuracy and F1, compared to using normal phrases. This is due to the improvement in generalization, which

takes place when synonymous phrases sharing similar meanings are collapsed together. It can also be observed that, using a domain lexicon whose entries constitute 80% of the total count of nouns and verbs in the corpus yields highest results regardless of the type of phrases being used. The 80% threshold provides results that are consistent with the previous experiment, and hence will be used later when evaluating FRN with the full HRS model.

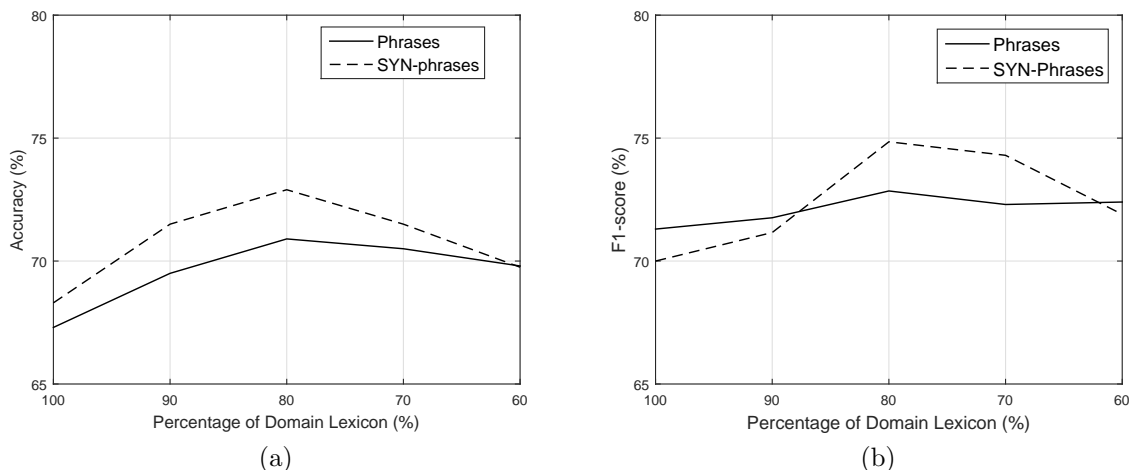


Figure 4.5: Evaluating the impact of synonymy when introducing SYN-phrases vs. phrases under different sizes of the domain lexicon. Figure (a) shows the accuracies, while Figure (b) shows the F1 scores.

### Comparison between FRN Standalone versus “FRN+HRS”

We evaluate the FRN method with and without the proposed HRS. First, we extract the different types of semantic and syntactic FRN features from the train set as follows:

- $n$ -word and  $n$ -characters: kfNgram [132] was used to extract 1, 2, 3-grams.
- $n$ -POS features: After POS tagging the reviews using Stanford tagger [133], kfNgram was used to extract 1, 2, and 3-tags.
- $n$ -POSword features: To extract these features, the tagged reviews were fed as input to kfNgram. Each tag and its corresponding word were considered as a single token. Sequences of consecutive 1, 2, and 3-POSwords were then extracted.
- $n$ -legomena features: The  $n$ -grams extracted in the step above were used by replacing any once- and twice-occurring words by “HAPAX” and “DIS”,

respectively. For example, if the word *Titanic* is only found once in the entire corpus, and an extracted 2-gram was *Titanic rocks*, the word *Titanic* is then replaced by “HAPAX” to form the 2-legomena feature *HAPAX rocks*.

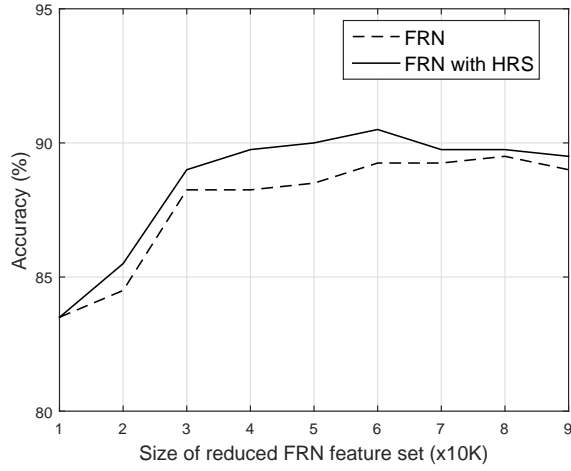
- *n*-semantic features: After replacing all words in the *n*-gram features by their corresponding synonym group (SYN#) semantic labels, kfNgram was used to generate the counts for each synonym group.
- Information Extraction Patterns (IEP): extracted using the Sundance package [123].

Second, we develop the proposed notions features as follows. After developing the domain lexicon with size of 80%, and selecting SYN-phrases to represent the textual characteristics of the notions, we proceed to develop the notions database by assigning a sentiment score to each SYN-phrase. The sentiment score of a SYN-phrase is defined as the difference in PMI scores calculated between that phrase and each of the sentiment classes, as illustrated in Equation (4.1). A score greater than 0 implies that the notion has a positive sentiment, otherwise it is negative. Once the notions database is developed, each notion in a review is assigned a working memory weight that is based on its location in that review, as shown in Equation (4.2).

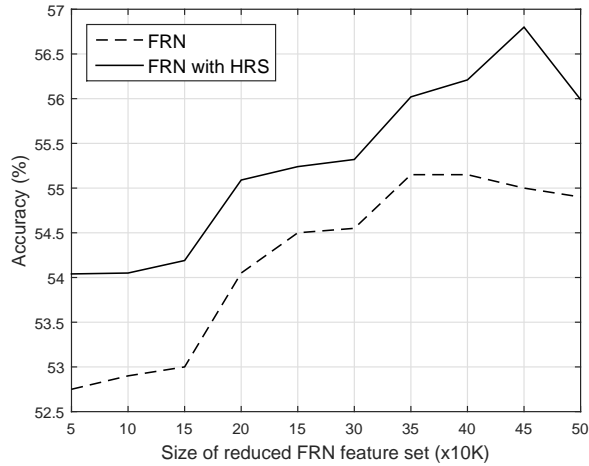
To reproduce FRN experiments, the FRN algorithm is first applied to reduce the extracted FRN features. The algorithm is set to output several reduced feature sets. For the IMDB corpus, the size of the reduced feature sets ranged between 10K and 100K features. For the YELP corpus, the sizes ranged between 50K and 500K since the corpus is bigger than IMDB with many more features to be extracted. The top-ranked selected features are then used to train the SVM model. To evaluate FRN with the proposed HRS model, we fuse the notions features with the different sets of reduced FRN features. Each notion is represented by the product of its sentiment score and its working memory weight in every document. The sentiment classification results in Figure 4.6 show that applying HRS with the notions features, introduces consistent improvements in accuracy on both datasets. On IMDB, and despite the existing high accuracy with FRN standalone close to 90%, HRS pushed the accuracy even higher with an average of 1%. Similarly on YELP, HRS improved the accuracy by 1.07% on average. It is worth mentioning that these performances can be further improved with optimization of each of the added HRS steps.

### 4.4.3 Evaluating GRNN with HRS

In this subsection, we evaluate the impact of applying the HRS model to GRNN. As shown in subsection 4.3.2, GRNN does not model the “*situation model of interpretation*”. To address this gap with HRS, we proposed a sentiment embedding



(a)



(b)

Figure 4.6: Evaluating the FRN method with and without HRS under different sizes of reduced features. (a) shows results on the IMDB corpus. (b) shows results on the YELP corpus.

technique to model this aspect by deriving word embeddings that encode sentiment information, and use them along with traditional embeddings that capture syntactics and semantics.

In the original GRNN model, words are represented with 200-dimensional embedding vectors derived using word2vec [64]. In the proposed setup with HRS, each word will be represented using the concatenation of two 100-dimensional embedding vectors. The first part is derived using word2vec to capture syntactic and semantic properties of the word, and the other part is derived using the proposed sentiment embedding to encode the word’s sentiment. Both vectors are

obtained using the training set only to avoid overfitting.

We used SentiWordNet lexicon [21] as the source of word-level sentiment information to pre-train the sentiment vectors. In order to derive sentiment vectors for as many words as possible, the lexicon is extended by including stopwords, punctuation, digits and out-of-vocabulary (OOV) words. Stopwords, punctuation and digits are assigned an objectivity score equal to 1 and positivity/negativity scores equal to 0. On the other hand, each OOV word is assigned scores that are equal to the average scores of all SentiWordNet words that co-exist with it in the same sentence.

After tuning the GRNN parameters using the training and the development sets, as done in [13], each model is evaluated on the test set. To minimize bias in the results, the experiments are repeated 50 times with different random choices of initialization weights. Accuracies and class-level F1 scores are averaged over the 50 rounds and are illustrated in Table 4.3.

Corpus	Approach	Accuracy	F1-score					avg.
			very neg	neg	neut	pos	very pos	
YELP	GRNN	<b>57.7</b>	65.5	36.7	36.0	71.1	30.6	<b>48.0</b>
	GRNN+HRS	<b>60.2</b>	67.9	39.2	39.3	73.2	34.9	<b>50.9</b>
IMDB	GRNN	<b>90.8</b>	–	90.9	–	90.8	–	<b>90.8</b>
	GRNN+HRS	<b>92.0</b>	–	92.3	–	91.2	–	<b>92.1</b>

Table 4.3: Results obtained on both datasets (YELP with 5 classes and IMDB with 2 classes) using GRNN with and without HRS.

Results illustrated in Table 4.3 show that applying HRS to GRNN results in performance improvement on both datasets. On YELP, adding HRS introduced 4.74% relative improvement in accuracy and 5.8% relative improvement in average F1 score. These are significant improvements given the challenge of 5-way sentiment classification. On IMDB, adding HRS introduced 1.31% relative improvement in accuracy and 1.43% relative improvement in average F1 score.

Table 4.4 summarizes the results for comparison of the HRS success with FRN versus GRNN to evaluate the impact of the HRS meta-framework on machine learning approaches based on feature engineering versus deep learning, respectively. For FRN-related experiments, we report the average performance over different sizes of reduced feature sets. For GRNN-related experiments, we report the average performance across different rounds with different choices of weight initialization. Results in Table 4.4 show that the HRS frameworks resulting from each method yield improvement compared to the methods without HRS. It can also be observed that HRS produced more improvement to GRNN in comparison

to FRN. One reason for this observation is that we relied on feature engineering with some heuristics, such as notions sentiment scoring and memory weighting scheme, to address the HRS-related gaps in FRN. On the other hand, in GRNN, optimization formulations were used to embed sentiment information and assign memory weights. It is important to note that, even with sub-optimal approaches to address HRS gaps in FRN, the resulting framework achieved a better performance.

<b>Approach</b>	<b>IMDB</b>		<b>YELP</b>	
	Accuracy	Avg. F1-score	Accuracy	Avg. F1-score
FRN	87.9	87.8	54.2	40.9
FRN+HRS	88.7	88.6	55.3	41.6
GRNN	90.8	90.8	57.7	48.0
FRN+HRS	92.0	92.1	60.2	50.9

Table 4.4: Summary of all results obtained by applying HRS to FRN and GRNN approaches on the IMDB corpus (5 classes) and the YELP corpus (5 classes)

In the next two chapters, we present particular solutions to improve the inference step of the meta-framework, with special focus on applicability to Arabic language.

# Chapter 5

## Recursive Auto Encoders for OMA

In this chapter, we explore the use of the Recursive Auto Encoder (RAE) [33] for opinion mining in Arabic, with focus on addressing challenges presented by: 1) lexical sparsity and ambiguity of Arabic, 2) inaccurate word embeddings, and 3) sub-optimal order of composition. To address these limitations with RAE, we train the model with morphologically-tokenized text to reduce language sparsity. We derive word sentiment embeddings to improve the model’s input. Finally, we use automatically-generated syntactic trees to improve composition. We describe the baseline RAE model for opinion mining in section 5.1, and then present solutions to address each of the above-mentioned challenges in section 5.2. Experimental results are presented in section 5.3 and show that each of the proposed solutions introduces significant performance improvements compared to the baseline RAE model.

### 5.1 RAE for Opinion Mining

The Recursive Auto Encoder (RAE) model that was proposed for sentence-level opinion mining [33] is composed of two stages, as illustrated in Figure 5.1. The first stage is unsupervised; it derives a vector (embedding) for each sentence by applying an auto encoder (AE) to its constituents, to at a time, in a recursive manner. The second stage is supervised; it uses the derived sentence embeddings to train a logistic regression (softmax) sentiment classification.

The AE is trained one sentence at a time, and each word in the sentence is represented by a  $d$ -dimensional vector that is derived using some neural language model (NLM). The word vectors are dense and low-dimensional, compared to the commonly-used BoW, thus eliminating sparsity issues. Raw words are transformed into word vectors using a lookup table  $L \in \mathbb{R}^{d \times V}$  that contains a  $d$ -dimensional vector for each word in the language vocabulary of size  $V$ . Results

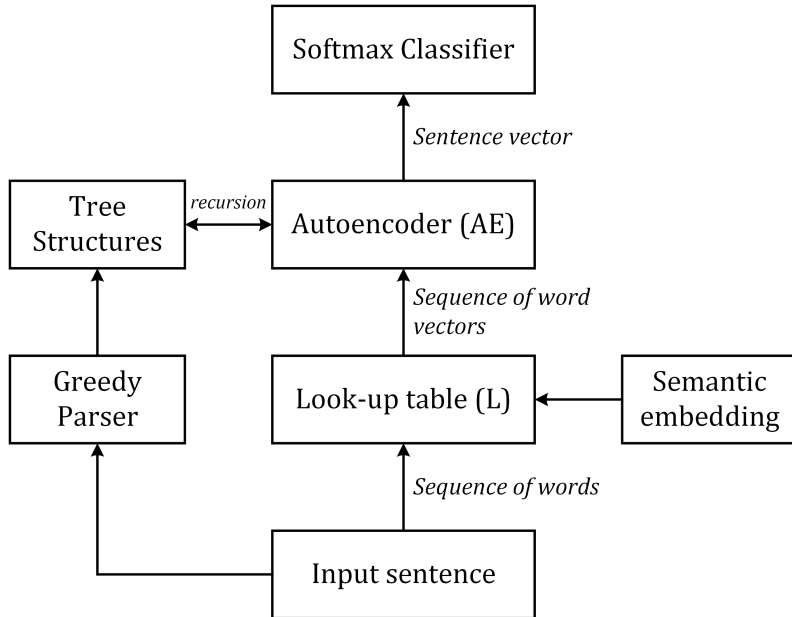


Figure 5.1: The framework of RAE, which we refer to as baseline RAE.

reported by [33] indicate that using randomly-initialized word vectors achieved very similar performances to the case of using word vectors that are pre-trained using the C&W model [99].

RAE performs composition recursively following the structure of a binary parse tree, in which leaf nodes correspond to words, non-terminal nodes correspond to intermediate constituents, and the root node corresponds to the full sentence. At each step of recursion, the AE takes two input vectors  $x_1, x_2 \in \mathbb{R}^d$  and produces an output ‘parent’ vector  $o$ , as shown in Equation (5.1), where  $f$  is an element-wise nonlinearity function (usually ‘tanh’) The output  $o$  is fed again to the AE block along with the vector of the next node in the tree. It is also used to reconstruct the input vectors  $x'_1, x'_2 \in \mathbb{R}^d$ , as shown in Equation (5.2). This process continues, for each sentence, until an output vector  $o^*$  is produced at the root node, corresponding to the sentence embedding.

$$p = f(W_1^T [x_1; x_2]) \quad (5.1)$$

$$[x'_1; x'_2] = f(W_2^T o) \quad (5.2)$$

Figure 5.2 illustrates the structure of the AE along with its parameters.

The AE parameters  $\theta = \{W_1, W_2\}$  are updated to optimize its ability to reconstruct its input word vectors, or equivalently to minimize the reconstruction error  $E_{rec}$  shown in Equation (5.3).



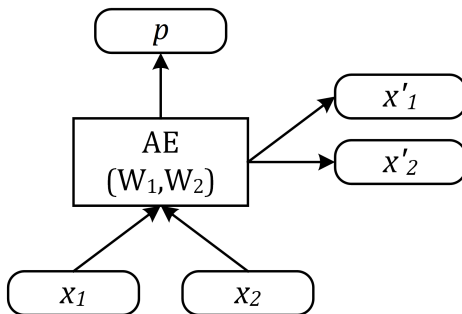


Figure 5.2: The structure of the AE (parameters, inputs and outputs).

$$E_{rec} = \| [x_1; x_2] - [x'_1; x'_2] \|^2 \quad (5.3)$$

The objective of training the AE is to obtain the parameters  $\theta^*$  that minimize the reconstruction error for all sentences. The reconstruction error  $E_{rec}$  is calculated at every step in the sentence, and the model's parameters are updated to minimize the overall reconstruction error, shown in Equation (5.4), where  $(\ell - 1)$  is the number of parsing steps for a sentence of length  $\ell$ . The solution is derived using stochastic gradient descent (SGD), and the overall  $E_{rec}$  is back-propagated down to the input layer in order to fine-tune the word vectors.

$$\psi^* = \arg \min_{\psi} \sum_{i=1}^{\ell-1} E_{rec,i} \quad (5.4)$$

The structure of the parse tree is discovered using an unsupervised “greedy” algorithm. For each sentence, given its word vectors  $X = \{x_1, \dots, x_\ell\}$ , this algorithm applies the AE to all adjacent word vector pairs  $[x_i; x_{i+1}]$ ,  $\forall x_i \in X$ . The pair with the minimum  $E_{rec}$  is added to the parse tree, and is replaced in the word list  $X$  by its AE output  $o$ . This process repeats until the whole input sequence is traversed and the sentence tree structure is obtained. Once all sentences are parsed, the AE parameters are updated to minimize the overall  $E_{rec}$  as shown in Equation (5.4). The final AE model is obtained after processing all sentences in the training set.

Sentence embeddings  $o^*$  are then used to train a supervised softmax layer to predict the class distribution for each sentence. Given  $K$  sentiment classes, the softmax layer produces, for each sentence, a  $K$ -dimensional multinomial distribution  $y \in \mathbb{R}^K$  as shown in Equation (5.5).

$$y = \frac{1}{1 + \exp(-W_s^\top o^*)} \quad (5.5)$$

where  $o^* \in \mathbb{R}^d$  is the sentence embedding, and  $W_s \in \mathbb{R}^{d \times K}$  is the set of softmax parameters. The  $k^{th}$  element in  $y$  is interpreted as the conditional probability

of the  $k^{th}$  class given the sentence representation. The set of parameters  $W_s$  are obtained by minimizing the cross-entropy error, shown in Equation (5.6), using SGD. This is the only supervised learning step in the RAE sentiment model.

$$E_{ce}(o^*, y; W_s) = - \sum_{k=1}^K t_k \log y_k \quad (5.6)$$

where  $t_k$  is the gold conditional probability of the  $k^{th}$  class given sentence representation  $o^*$ . For binary classification,  $t \in \mathbb{R}^2$  is  $[1, 0]$  for class 1 and  $[0, 1]$  for class 2.

## 5.2 RAE for Opinion Mining in Arabic

In this section, we present a framework that uses RAE to model composition, and that addresses the following challenges to improve opinion mining in Arabic: 1) lexical sparsity and ambiguity of Arabic, 2) inaccurate word embeddings, and 3) sub-optimal order of composition. This framework is illustrated in Figure 5.3, with solutions highlighted in blue color. In particular, we perform morphological tokenization to overcome morphological richness and lexical sparsity and ambiguity. We use a neural network architecture to derive embeddings that capture word-level sentiment aspects. Finally, we use syntactic parsers to generate grammatically-motivated trees to improve composition. These solutions are described with further details in the following subsections.

### 5.2.1 Tokenization to address Sparsity and Ambiguity

Arabic is characterized by morphological richness leading to lexical sparsity, where inflectional variations of the same word share the same core meaning, but have different forms. This affects the ability of machine learning models to generalize to new, unseen text. Furthermore, the rich concatenative morphology leads to lexical ambiguity, where words of the same form have different morphology leading, in some cases, to different meanings and sentiments.

To overcome these challenges, we train the recursive deep models using input text that is morphologically-tokenized, where words are further split into more basic morphemes. We perform tokenization following the ATB scheme [134], where all clitics except for the definite article *ال* *Al* ‘the’ are separated from the stem.

Reducing words to their basic morphemes improves the ability of the model to generalize to new, unseen morphological variations of the training instances, as such variants tend to share the same base words identified by tokenization. For example, models trained to learn the semantics of *لَيَنْتَصِرَ* *liyantaSir* ‘to win’, can use this knowledge to understand a new word *وَسَيَنْتَصِرَ* *wasayantaSir* ‘and he

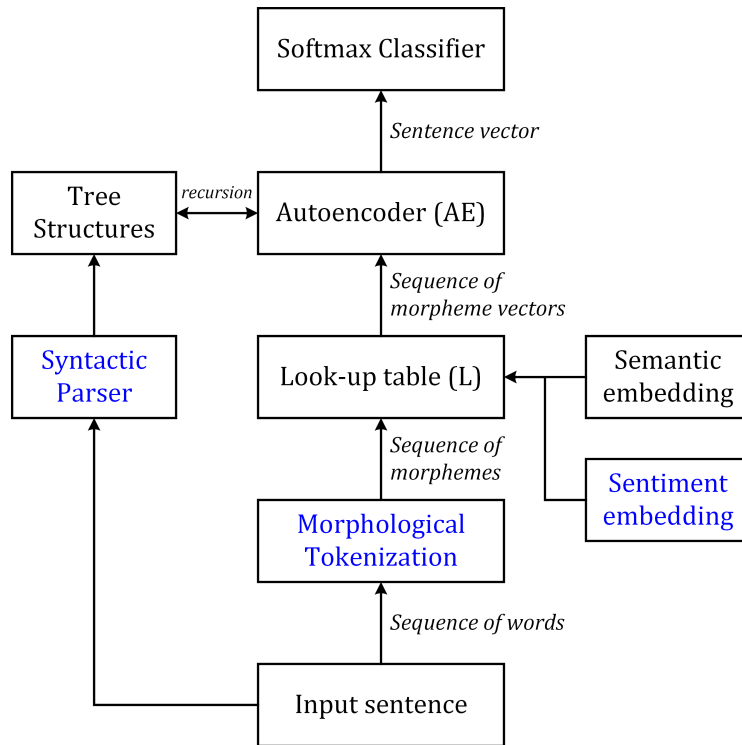


Figure 5.3: The framework of RAE for opinion mining in Arabic, with solutions highlighted in blue.

will win’ because both words share the same base *ينتصر* *yntSr* that is identified via tokenization. Furthermore, tokenization disambiguates words that share the same form but differ in their morphology, such as *بِشَرِّهَا* *bi+\$ar~i+haA* ‘by her evil’ and *بَشَّرَهَا* *ba\$~ra+haA* ‘told her good news’.

### 5.2.2 Sentiment Embedding to improve Word Embeddings

Word embeddings that are generated using existing NLMs do not capture sentiment aspects of the word, and hence composition models will produce sentence embeddings that do not capture the sentiment content of the text. Moreover, the approach to derive the word embeddings is based on co-occurrence statistics, thus similar embeddings are derived to words that occur in similar contexts, even if they express different sentiments. Using such embeddings to train composition models will generate inaccurate sentence representations that affect sentiment classification performance.

To overcome this problem, we propose to derive embeddings that capture

a broader range of the words’ semantics, with sentiment included. Inspired by the C&W embedding model [99], we present a model to perform word sentiment embedding, leveraging existing sentiment lexicons. The combination of semantic embeddings (C&W) and sentiment embeddings are then used to derive sentence embeddings that are more complete in their representation.

## Semantic Embedding

The C&W model generates word vectors via supervised training of a  $n$ -gram validity classifier over a large unlabeled corpus.  $N$ -grams are automatically labeled as follows;  $n$ -grams that exists in the corpus are considered ‘valid’, and by randomly changing one of the  $n$ -gram’s words, we obtain an ‘invalid’  $n$ -gram. Word  $n$ -grams are then transformed into vector  $n$ -grams using a randomly-initialized lookup table  $L$ , and are used to train a softmax classifier to discriminate between valid and invalid  $n$ -grams. Classification errors are back-propagated to the input layer to update the word vectors in  $L$ . The resulting vectors capture semantic and syntactic aspects of the words [99]. For simplicity, we refer to this process as “semantic embedding”.

In this dissertation, we introduce an unsupervised stage to pre-train the lookup table  $L$  before using it for validity classification. This stage is shown as “Stage 1” in Figure 5.4. Word  $n$ -grams in the corpus are transformed into vector  $n$ -grams using a randomly-initialized lookup table  $L$ , and are used to train a Deep Belief Network (DBN) generative model by stacking Restricted Boltzmann Machines (RBM). During this stage of pre-training, the reconstruction error is back-propagated to the input layer in order to fine-tune the word vectors in  $L$ . At the end of pre-training, the updated vectors in  $L$  are used to initialize the lookup table for use in the validity supervised learning (Stage 2 in Figure 5.4). Such initialization should improve validity classification by providing word vectors that reflect their local context in the  $n$ -grams.

## Sentiment Embedding

We adopt the same two-stage semantic embedding approach to capture sentiment-related information in the word vectors. The main difference lies in the objective of the supervised learning (stage 2); instead of predicting the validity of word  $n$ -grams, we predict the sentiment polarity of individual words. To train the classifier towards this objective, we use examples extracted from existing sentiment lexicons. In particular, we use the large-scale Arabic Sentiment Lexicon (ArSenL) [42] to obtain the words’ sentiment polarities. Consequently, the constructed lookup table  $L$  contains vectors reflecting word-level sentiment aspects.

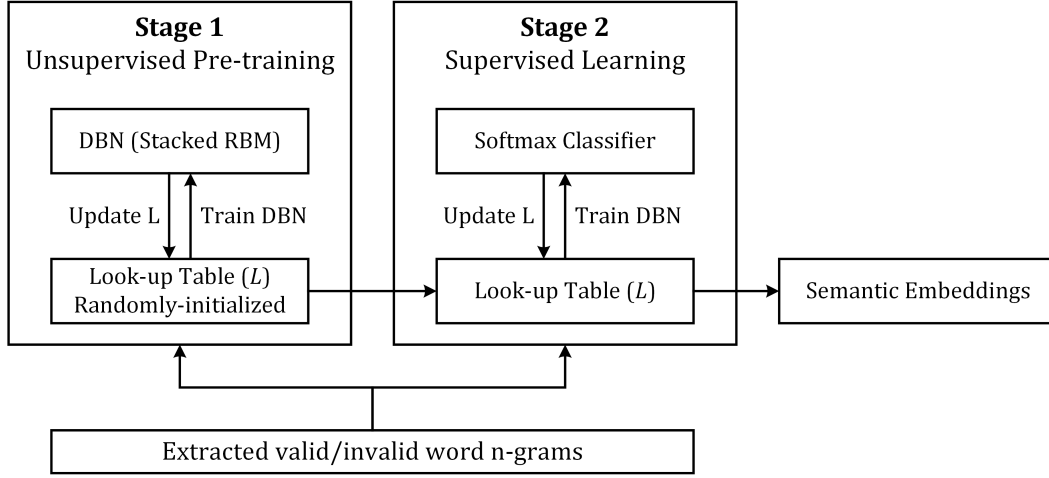


Figure 5.4: The architecture of the C&W embedding model, extended with pre-training stage.

### Fusing Sentiment and Semantic Embeddings

To fuse both types of the acquired word embeddings, the RAE composition model is used, independently, to generate two sentence representations,  $o_{sem}^*$  and  $o_{senti}^*$ , using the semantic and the sentiment word embeddings, respectively. Then, the “complete” sentence embedding,  $o_{complete}^*$ , is formed by concatenating both representations, and used to train the softmax sentiment classifier. Given  $K$  sentiment classes, the softmax layer produces a probability distribution  $y \in \mathbb{R}^K$ , in which the  $k^{th}$  element corresponds to the conditional probability of the  $k^{th}$  class given  $o_{complete}^*$ . This probability can be modeled as a Bayes Network as shown in Equation (5.7).

$$p(k|o_{complete}^*) = p(k|o_{sem}^*, o_{senti}^*) = \frac{p(o_{sem}^*, o_{senti}^*|k) p(k)}{p(o_{sem}^*, o_{senti}^*)} \quad (5.7)$$

Since the sentence representations for each type of embedding were generated independently, then Equation (5.7) can be simplified as follows:

$$p(k|o_{complete}^*) = \frac{p(o_{sem}^*|k) \cdot p(o_{senti}^*|k) \cdot p(y_k)}{p(o_{sem}^*) \cdot p(o_{senti}^*)} = \frac{O(k|o_{sem}^*) \cdot p(k|o_{senti}^*)}{p(k)} \quad (5.8)$$

where the term  $p(k)$  can be regarded as a normalization factor that can be easily obtained for the sentiment distribution in the annotated training data. Each of the remaining terms can be obtained by training the model with the corresponding type of word embeddings.

It is worth mentioning that the word embeddings are derived according to the tokenization scheme being experimented with. In other words, if raw input

is used, then embeddings are learned for the raw words. Similarly, if input is morphologically-tokenized, then embeddings are learned for the tokenized words.

### 5.2.3 Syntactic Parsing to improve Composition

The greedy parsing algorithm does not capture information about the language syntax and its grammatical structure. Hence, generated trees are not optimal in the context of combining semantically and syntactically-related constituents, which affects the ability of the recursive deep models to capture compositional semantics in text.

To overcome this limitation, we use syntactic parse trees that reflect the phrase structure of the sentence. We utilize the Stanford lexicalized parser [135] to automatically generate the syntactic parse trees, over which the AE will be recursively trained. The parser requires the input text to be morphologically-tokenized according to the ATB scheme [134], which is consistent with our choice of tokenization described in subsection 5.2.1 to address sparsity and ambiguity issues. Using syntactic parse trees to define the order of composition should improve the derived sentence embeddings, since the path for AE recursion is now consistent with the grammatical rules, and reflects the natural order in which constituents are combined to express the overall meaning of the sentence.

It is worth mentioning that syntactic parse trees are not necessarily binary, and therefore cannot be used to train recursive models that require inputs and outputs with consistent dimensionalities. Therefore, we transform the grammar of the obtained parse trees' to the Chomsky Normal Form (CNF) [136] using left-factoring, where the choice of left (vs. right) was made such that the model's recursion is consistent with the direction readers follow to combine words while reading Arabic text. The CNF grammar contains only unary and binary production rules. By collapsing the unary productions, we obtain binary parse trees that can be used to train recursive models.

## 5.3 Experiments and Results

In this section, we evaluate the different solutions, separately and combined, against the baseline RAE model [33]. We also compare the full RAE framework (with all solutions combined) against several well-known Arabic sentiment models from the literature. We focus on the common task of binary sentiment classification, where the sentiment class can be either positive or negative.

### 5.3.1 Datasets and Experimental setup

Evaluation is conducted using three Arabic corpora that represent different genres and that use different Arabic writing styles. Table 5.1 illustrates the size and

sentiment distribution for these corpora.

The first corpus was developed by [80]. It consists of newswire articles written in MSA, extracted from the ATB Part 1 V 3.0, and annotated at the sentence-level with one of the following classes: positive, negative, neutral and objective. Out of 2,855 sentences, we only used 1,180 sentences with either positive or negative sentiments. We refer to this corpus as *ATB*.

The second corpus is a set of tweets collected and annotated by [137]. Tweets usually combine different writing styles including MSA, a wide variation of dialects, URLs, images and significant amounts of misspellings due to the tweet length restriction. Similar to *ATB*, we only used 2,311 tweets with either positive or negative sentiments, and we refer to this corpus as *Tweets*.

The third corpus is composed of Initially, 1,177 comments that were initially extracted by [138] from the Qatar Arabic Language Bank (QALB) [139] to create a corpus for opinion target identification. Comments are written in MSA, but also contain misspellings and dialectal Arabic (DA). We annotated this corpus at the phrase and the comment-level using CrowdFlower, where each text was assigned one of five sentiment labels: very negative, negative, neutral, positive and very positive. We only used comment-level annotations, selecting only those (1,133 comments) with either positive or negative sentiments, ignoring the sentiment intensity. We refer to this corpus as *QALB*.

Dataset	Corpus size	Positive (%)	Negative (%)
ATB	1,180 sentences	68.7%	31.3%
Tweets	2,311 tweets	58.4%	41.6%
QALB	1,133 comments	34.8%	65.2%

Table 5.1: Characteristics of the different evaluation corpora.

Each corpus was preprocessed with the purpose of cleaning and improving the representation of the input data. Preprocessing included: 1) removing non-Arabic words, 2) segmentation by separating punctuation and digits from words, and 3) normalization. The main purpose of normalization is to improve the quality and coverage of the word embeddings, and to provide proper input to the parser for accurate parsing. We applied normalization to *characters* by normalizing repeated characters in elongated words, to *emoticons* by replacing emoticons with global ‘happy/sad’ special tokens using a manually-compiled list of emoticon shortcuts, and to *parentheses* by normalizing the different forms of parentheses into one form (square brackets). To evaluate the impact of normalization, we trained several baseline RAE models using different versions of the inputs, each reflecting a specific type of normalization. We used the *Tweets* dataset for this

experiment as it contains significant amounts of elongated words and emoticons, compared to the other corpora. Results indicate that applying *character*, *emoticon* and *parenthesis* normalization improved the classification accuracy by 0.7%, 0.8% and 0.8%, respectively, compared to the case of “no normalization”. Furthermore, the combination of all types of normalization improved accuracy by 2.2%, and thus will be used in further experiments.

Twitter-specific preprocessing was applied to the *Tweets* dataset and included: removing user mentions, re-tweet (RT) labels and URLs, and also preprocessing hashtag mentions by removing the hashtag symbol and the ‘under-scores’ connecting between multiple words in a single tag. These techniques are common practice in the literature [140–142].

The performance is quantified using the accuracy and the F1-score averaged over both positive and negative classes. To ensure statistical significance of results, the different models are evaluated using 10-fold cross-validation. The AE is formed of three layers, and both the size of the word embeddings and the number of hidden neurons in each layer are set to 50, which yield the best results in a preliminary experiment on a separate dataset.

Software and tools that were used to conduct the required experiments are mentioned in Appendix A.

### 5.3.2 Ablation Analysis

We evaluate the improvements achieved by each of the proposed solutions. Tables 5.2, 5.3 and 5.4 illustrate the impact of applying 1) morphological tokenization to combat sparsity and ambiguity, 2) sentiment embedding to provide better input word vectors and 3) syntactic parsing to allow better composition. We compare against the baseline RAE model [33] that uses randomly-initialized vectors for raw words, and that iterates over trees derived using the greedy parser.

#### Impact of Morphological Tokenization

Results indicate that tokenization yields significant improvements over the baseline RAE, and for all datasets. Improvement is mainly due to reduction in both lexical sparsity and ambiguity, where raw words are rendered to their base words, allowing the model to generalize to new unseen words. Results also highlight the added benefit of performing composition at a finer-level of morphology instead of raw words that are packed with many information.

#### Impact of Sentiment Embedding

Experiments were conducted to evaluate the impact of using sentiment versus using semantic embedding, and also to evaluate the impact of fusing both embeddings. Since we are comparing to the baseline RAE, experiments are carried



<b>Experiments on ATB</b>	<b>Accuracy</b>	<b>F1-score</b>
Baseline RAE	74.3	73.5
Morphological Tokenization	77.3 (+3.0)	76.2 (+2.7)
Semantic embedding	76.5 (+2.2)	75.3 (+1.8)
Sentiment embedding	79.4 (+5.1)	78.7 (+5.2)
Both embeddings (concatenation)	81.9 (+7.6)	82.3 (+8.8)
Syntactic parsing	76.7 (+2.4)	76.1 (+2.6)
All (Improved RAE)	<b>86.5 (+12.2)</b>	<b>84.9 (+11.4)</b>

Table 5.2: Impact of each solution evaluated on the ATB corpus.

<b>Experiments on QALB</b>	<b>Accuracy</b>	<b>F1-score</b>
Baseline RAE	71.6	66.5
Morphological Tokenization	75.4 (+3.8)	71.6 (+5.1)
Semantic embedding	72.6 (+1.0)	68.4 (+1.9)
Sentiment embedding	73.8 (+2.2)	70.3 (+3.8)
Both embeddings (concatenation)	74.3 (+2.7)	71.1 (+4.6)
Syntactic parsing	71.0 (-0.6)	65.6 (-0.9)
All (Improved RAE)	<b>79.2 (+7.6)</b>	<b>75.5 (+9.0)</b>

Table 5.3: Impact of each solution evaluated on the QALB corpus.

on raw untokenized words. Results show that, for all datasets, using sentiment embedding outperformed semantic embedding. This indicates that sentiment is a much more important information that needs to be embedded into the word and sentence vectors given the task of sentiment analysis. Results also indicate that fusing both types of embeddings introduces additional improvements since the classifier is now trained using sentence embeddings that are more complete as they capture syntactic, semantic and sentiment information.

At last, it can be observed that using semantic embedding introduced significant improvements compared to using randomly-initialized word vectors, which does not align with findings in English, where semantic embedding only achieved marginal improvements [33]. This can be attributed to the unsupervised pre-

Experiments on Tweets	Accuracy	F1-score
Baseline RAE	69.7	61.1
Tokenization	70.9 (+1.2)	62.7 (+1.6)
Semantic embedding	71.3 (+1.6)	64.4 (+3.3)
Sentiment embedding	71.6 (+1.9)	65.8 (+4.7)
Both embeddings	73.8 (+4.1)	67.4 (+6.3)
Syntactic parsing	67.4 (-2.3)	58.1 (-3.0)
All (Improved RAE)	<b>76.9 (+7.2)</b>	<b>68.9 (+7.8)</b>

Table 5.4: Impact of each solution, evaluated on the Tweets corpus.

training stage that was introduced to the semantic embedding model. It can also be an indicator that Arabic words, as opposed to English words, are packed with many information that cannot be fully-captured by only fine-tuning randomly-initialized vectors during RAE training, but require better initialization using pre-trained embeddings.

### Impact of Syntactic Parsing

We evaluated the impact of using syntactic parse trees versus trees generated by the greedy parser. Since we are comparing to the baseline RAE, experiments are carried on raw untokenized words. Since the Stanford parser generates trees for morphologically-tokenized text, we detokenize the resulting trees by collapsing nodes that correspond to morphemes of the same word into one node.

Experiments results indicate that using syntactic parse trees improved the performance only on the ATB dataset, whereas it resulted in performance degradation on the other datasets. The main reason for this behavior is that ATB sentences are written in MSA and comply to the grammatical rules of the language, allowing the parser to generate trees that reflect the natural order in which words and constituents are combined in a sentence. On the other hand, the QALB and Tweets datasets contain significant amounts of noise represented by dialectal Arabic and misspellings, especially the Tweets dataset. This is reflected in the results, where automatic parsers failed to produce meaningful trees, causing performance degradation compared to the greedy parser.

To confirm the importance of using syntactic parse trees, we evaluated the impact of using gold syntactic trees that are manually-developed by expert linguists, and hence free of automatic parsing errors. Gold trees are only available for the ATB dataset since its sentences are extracted from the ATB Part 1 V 3.0.

Having access to V 4.1 [143], we align its trees with those in V 3.0, and then we obtain the gold trees for the sentences in the ATB dataset. We evaluated three different types of parse trees: those discovered using the greedy parser, those generated by automatic parsers (the Stanford parser) and the gold trees. Since the gold trees assume the input to be morphologically-tokenized, we evaluated the different trees following the same assumption. Results in Table 5.5 indicate that using the gold trees achieved the best performance, and that errors introduced by automatic parsing caused around 0.5% degradation on the overall performance, when used to train RAE models on MSA data.

	Accuracy	F1-score
Tokenization + greedy trees	77.2	76.2
Tokenization + automatic Stanford trees	78.5 (+1.3)	77.8 (+1.6)
Tokenization + gold PATB trees	78.9 (+1.7)	78.1 (+1.9)

Table 5.5: Impact of different tree structures on the performance of RAE for opinion mining in Arabic.

### Combining All Solutions

The full RAE framework for opinion mining in Arabic is achieved by implementing all solutions that address the different challenges. Tables 5.2, 5.3 and 5.4 show that this framework achieves highest performance compared to its individual components. Compared to baseline RAE, it achieved absolute accuracy improvement of 12.2%, 7.6% and 7.2% on the ATB, QALB and Tweets datasets, respectively. These improvements, compared to the baseline RAE, are statistically significant with 90% confidence at 9 degrees of freedom.

The highest improvement was achieved on the ATB dataset since it consists of sentences written in MSA, hence benefiting from all proposed contributions including the use of syntactic parse trees, as opposed to the other datasets where syntactic parsing did not achieve the hoped-for performances. Results also indicate that the different solutions are synergetic. For example, tokenization helps producing better embeddings as it reduces the language sparsity and resolves much of the morphological complexity that becomes explicitly highlighted at the surface (form)-level instead of being embedded into the learned representations.

### 5.3.3 Results Benchmarking

In this section, we evaluate the RAE framework for OMA against several Arabic sentiment models proposed in literature. We compare to SVM and NB models

trained using BoW features with different choices of preprocessing and feature representation [38, 75]. We train using word, stem and lemma  $n$ grams ( $n = 1, 2, 3$ ), represented using presence, term frequency (TF) and term frequency-inverse document frequency (TFiDF) scores. We report the best results achieved using the TFiDF scores. We also train SVM using aggregated sentence-level sentiment scores based on the ArSenL sentiment lexicon as proposed by [42]. This model achieved better results than [80] on the ATB dataset. At last, we compare to several deep learning models evaluated for OMA. These models are: Deep Belief Networks (DBN), Deep Neural Networks (DNN), and Deep Auto Encoders (DAE) combined with DBN. These models are trained using two types of features: the BoW and the sentence-level aggregated sentiment scores. All the above-mentioned models are also trained using 10-fold cross-validation to ensure statistical significance of the results. Table 5.6 illustrates the performance improvement achieved by our solutions, compared to the aforementioned sentiment models. Comparing to the second-best approach, our improved RAE model introduces accuracy improvements of 7.3%, 1.7% and 7.6% on the *ATB*, *QALB* and *Tweets* datasets, respectively.

Model	Features	ATB		QALB		Tweets	
		accuracy	F1-score	accuracy	F1-score	accuracy	F1-score
DNN	ArSenL scores	54.7	43.9	52.3	48.9	58.3	50.7
	BoW	39.3	38.8	43.6	40.1	54.6	38.9
DBN	ArSenL scores	56.9	46.2	55.4	47.5	61.2	54.5
	BoW	40.9	39.7	45.0	42.3	57.6	43.2
DAE-DBN	ArSenL scores	59.7	59.9	59.2	54.2	63.7	57.8
	BoW	42.9	43.3	47.5	44.6	59.3	44.6
linear SVM	ArSenL scores	62.8	56.7	71.0	62.8	68.7	40.7
	word 1-grams	75.3	73.9	76.1	71.3	62.1	54.7
	stem 1-grams	77.5	76.6	77.5	74.7	62.4	55.9
	lemma 1-grams	77.5	76.5	77.1	74.7	62.9	56.7
	word 1-2-grams	76.2	73.9	73.3	62.3	68.5	56.6
	stem 1-2-grams	79.2	77.7	77.4	70.3	68.4	57.4
	lemma 1-2-grams	78.7	77.2	76.9	69.9	68.7	57.8
	word 1-3-grams	75.3	71.8	69.9	54.0	68.5	54.5
	stem 1-3-grams	77.5	75.2	74.4	63.9	69.3	56.7
	lemma 1-3-grams	79.1	77.1	74.5	64.4	68.7	55.7
NB	word 1-grams	69.8	69.4	69.5	65.7	54.7	53.5
	stem 1-grams	74.4	73.9	70.6	66.1	56.3	54.3
	lemma 1-grams	73.6	73.2	68.9	65.1	55.2	53.5
	word 1-2-grams	70.1	69.3	72.4	67.9	56.7	55.0
	stem 1-2-grams	73.8	73.0	73.3	67.8	57.9	55.3
	lemma 1-2-grams	74.2	73.4	71.5	65.7	56.0	53.8
	word 1-3-grams	70.3	69.5	72.6	68.2	56.8	55.2
	stem 1-3-grams	73.6	72.8	73.4	67.9	58.3	55.6
	lemma 1-3-grams	73.1	72.1	72.2	66.4	56.0	53.8
RAE	raw words	74.3	73.5	70.8	65.0	69.7	61.1
RAE for Arabic	tokenized words	<b>86.5</b>	<b>84.9</b>	<b>79.2</b>	<b>75.5</b>	<b>76.9</b>	<b>68.9</b>

Table 5.6: Results of benchmarking the performance of RAE for OMA against sentiment models from the literature.

## Chapter 6

# Morphologically-Enriched Recursive Deep Models

In this chapter, we explore the Recursive Neural Tensor Networks (RNTN) [34] for OMA, with focus on addressing challenges presented by: 1) lexical sparsity and ambiguity of Arabic, and 2) lack of sentiment lexical resource to train RNTN. To address these limitations, we exploit the orthographic and morphological space to enrich the RNTN with useful information. We also present the Arabic sentiment treebank (ARSENTB), the first of its kind for Arabic. The baseline RNTN model for opinion mining in section 6.1, and then present solutions to address each of the above-mentioned challenges in sections 6.2 and 6.3 Experiments and results are presented in section 6.4 and show that, compared to the baseline RNTN, morphologically-enriched RNTNs achieve significant accuracy improvements up to 8.2% and 9.4% at the phrase and the sentence-level, respectively. They also outperform SVM and RAE classifiers trained with similar morphological considerations. We also present a qualitative analysis of the recursive deep models for OMA with respect to the human reading for sentiment (HRS) meta-framework, introduced earlier in chapter 4.

### 6.1 RNTN for Opinion Mining

The Recursive Neural Tensor Networks (RNTN) model follows a similar approach to RAE; it uses a composition model recursively over a parse tree to derive the sentence embedding, which is then used for sentiment classification [34]. However, RNTN differs from RAE in the following aspects. 1) It performs composition using a tensor-based function (tensor product). 2) Softmax classifier is trained on top of each node of the tree, not only the root node.

RNTN takes a sequence of  $d$ -dimensional word vectors that can be either randomly-initialized or pre-trained using some NLM. Equation (6.1) shows how two input vectors are combined using the tensor product to produce the represen-

tation of their parent node. This process is performed recursively in a bottom-up fashion.

$$o = f \left( \begin{bmatrix} x_1 & x_2 \end{bmatrix} V^{[1:d]} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} + W \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \right) \quad (6.1)$$

where  $o$ ,  $x_1$  and  $x_2 \in \mathbb{R}^d$  are the output and input vectors, respectively,  $f$  is the ‘tanh’ nonlinearity function,  $W \in \mathbb{R}^{d \times 2d}$  is the set of model parameters to be learned, and  $V \in \mathbb{R}^{2d \times 2d}$  is one slice of the tensor matrix, where each slice aims to capture a specific type of composition.

Given  $K$  sentiment classes, a softmax layer is trained on top of each node in the tree, and produces a  $K$ -dimensional distribution, in which every element corresponds to the conditional probability of a particular sentiment label given the vector of the current node. RNTN parameters ( $W, V, W_s$ ) are trained to minimize the cross-entropy between the predicted and the gold sentiment distributions, for all nodes of the tree.

Figure 6.1 illustrates how RNTN is applied to predict sentiment in a three-word phrase. At each non-terminal node, the composition function  $g(\cdot)$  derives its vector representation, which is then used to predict the sentiment of that node.

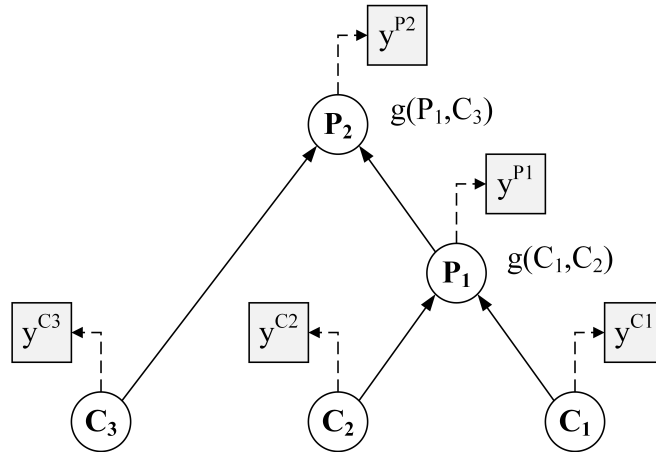


Figure 6.1: Sketch of RNTN applied to predict sentiment in a three-word phrase.

The RNTN model needs to be trained on a sentiment treebank; a collection of phrase structure parse trees with sentiment annotations at all levels of each tree. Such resource only exists for English; the Stanford Sentiment Treebank [34].

## 6.2 Addressing Rich Morphology & Ambiguity

A common solution in Arabic NLP, to overcome the challenges associated with morphological richness and ambiguity of Arabic, is to perform automatic in-context morphological analysis and disambiguation. The goal of this process is to identify for each word its exact “in-context” analysis that is represented by a set of inflectional features (voice, gender, part-of-speech, etc.), different levels of morphological segmentation and abstraction (morphemes, lemmas, stems, roots, etc.), in addition to orthographic features such as diacritization. While disambiguation reduces ambiguity, it does not address the question of lexical sparsity. This is why it is common to see efforts exploring the use of different morphological features to train machine learning models [40, 80, 94]. The typical solution to lexical sparsity is to operate at more abstract levels of morphology, with the following features being frequently explored in the literature:

- The **stem** (جذع) is defined as the part of the word that remains after deleting all clitics and affixes.
- The **lemma** is a conventionalized choice of one word to represent the set of all words related by inflectional (and not derivational) morphology.
- The **root** (جذر), in Arabic and other Semitic languages, is typically a sequence of three (sometimes four or five) consonantal radicals that abstracts away from all inflectional and derivational morphology [94]. Roots are often associated with highly abstract - and sometimes idiosyncratic - meanings that are shared in specific ways by specific lemmas, which may realize to the surface in different stems.
- The **ATB-token** refers to the result of performing morphological tokenization based on the ATB scheme [144] (splitting off all clitics except for the definite article).

For example, given the word *وَسَتَكْتُبُ* *wasataktub* ‘and you/she will write’, it corresponds to the lemma *كَتَبَ* *kataba*, the stem *كُتِبَ* *ktub*, the root *ك-ت-ب* *k-t-b* and the ATB-tokens *تَكْتُبُ* + *سَ* *wa+sa+taktub*.

Naturally, while these abstract forms reduce sparsity, they also introduce ambiguity. Roots are more ambiguous than lemmas and stems; and lemmas are more general than stems. For example, while the lemma *مَلْحَمَةٌ* *mlHmp* means ‘epic’, ‘butchery’ or ‘soldering shop’, its root *لحم* *lHm* includes additional meanings such as *لَحْمٌ* *laHm* ‘meat’, *لِحَامٌ* *liHAM* ‘soldering’, *لَحَّامٌ* *laH~Am* ‘butcher’, *التَّحَامُ* *AiltiHAM* ‘cohesion’, *لَحْمِيَّةٌ* *laHmiy~ap* ‘adenoids’, etc. Some stems may be similar



for different lemmas, which cause added noise. For example,  $قَائِم\ qA\}im$  is the stem from a number of words that belong to two lemmas:  $قَائِم\ qA\}m$  ‘existing’ and  $قَائِمَة\ qA\}mp$  ‘list’.

In the context of our work, we explore the morphological space of Arabic and enrich the recursive deep model with several levels of morphological abstraction. Then, we determine the level of morphology that achieves the best tradeoff between model sparsity and ambiguity for the task of OMA. We also explore the impact of diacritics on reducing lexical ambiguity at the different levels of abstraction.

## 6.3 The Arabic Sentiment Treebank

Previous work have showed the added benefits of using fine-grained sentiment prediction to train the composition model [34, 70]. However, training such models require a sentiment treebank, which only exists in English. In this section, we describe the system architecture to create ARSENTB; the first morphologically-enriched Arabic Sentiment Treebank that is needed to train RNTN for OMA under different morphological abstractions, as discussed in section 6.2.

### 6.3.1 System Architecture

Creating sentiment treebanks in MRLs is not straightforward. For instance, generating the syntactic parse trees requires automatic preprocessing such as normalization, tokenization and parsing, which all are prone to errors. As a result, annotators could end up annotating a corrupted version of the text that may not reflect the intended sentiment. For example, error in tokenization may cause a word like  $وَجَدَ\ waja\}da$  ‘he found’ to be tokenized as  $وَجَدَ + ا\ wa+jad\sim a$  ‘worked hard’ or ‘happened’, leading to different interpretations. Also, given our goal of annotating all phrases in the treebank, we need to provide annotators with very clear guidelines and instructions to ensure a proper handling of special cases such as phrases that lack context, or weird phrases that are obtained because of parsing errors, and so on.

Figure 6.2 describes the architecture of our proposed system to accurately and effectively create ARSENTB, while addressing the above-mentioned treebank-related challenges. This system is composed of the following components: morphological disambiguation, syntactic parsing, tree detokenization, sentiment annotation and morphological enrichment.

- **Morphological Disambiguation:** We perform morphological analysis and disambiguation in order to extract the in-context morphological features of each word. In particular, we extract the stem and the lemma

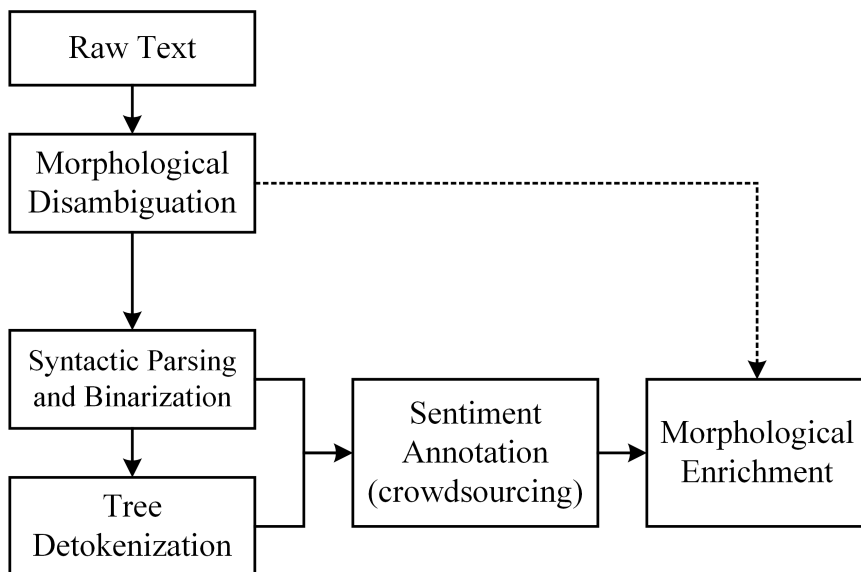


Figure 6.2: Architecture of the system to develop ARSENTB.

features, in addition to the predicted diacritics. These features will be used later on to enrich the generated treebank. We also perform Alef/Ya normalization and morphological segmentation according to the ATB scheme, in order to obtain the text representation that is required by typical phrase structure parsers. It is worth noting that character repetitions (elongations) are normalized as part of text preprocessing prior to morphological disambiguation. However, since elongations are useful for sentiment analysis [91], we preserve this information by marking words that originally-contained elongation.

- **Parsing and Binarization:** The ATB-tokenized text is fed to a trained parser [135] to generate phrase structure parse trees. These trees are not necessarily binary, and hence cannot be used to train recursive models that require inputs and outputs with consistent dimensionalities. Therefore, we use left-factoring to transform the trees to the Chomsky Normal Form (CNF) grammar that only contains unary and binary production rules. The choice of left (vs. right) was made such that sentiment composition follows the same direction readers follow to combine words when reading. After collapsing unary productions, we obtain binary trees that can be used to train recursive models.
- **Tree Detokenization:** As mentioned earlier, tokenization and parsing that are performed so far may produce errors that affect the quality and readability of phrases to be annotated. In order to provide annotators with phrases that look identical or as close as possible to the original text, we

perform tree ‘detokenization’ to obtain trees that represent the raw untok- enized text. Detokenization is done by merging leaf nodes corresponding to tokens of the same word, while preserving the trees’ binary structure. As a result, annotators will only see the raw text in phrases that are determined by a parser that ran on the tokenized form.

- **Sentiment Annotation:** We assign a sentiment label to each node (phrase) in the treebank to predict sentiment at all constituency levels. We designed a crowdsourcing sentiment annotation task using CrowdFlower. Before running the full task, we conducted a pilot study to tune the quality settings and get feedback on the clarity of the guidelines. Annotators were asked to assign each text one of five labels  $\{very\ negative, negative, neutral, positive, very\ positive\}$ . They were instructed not to be biased by their personal opinions, but rather reflect the authors’ opinions, similar to [145]. They were also asked to pay attention to linguistic phenomena such as elongation, emoticons and use of dialects. Annotators were provided with phrases that are randomly sampled from the treebank, so they are not affected by additional context. The model should be able to capture how different contexts affect sentiment.
- **Morphological Enrichment:** Our solution to improve recursive deep models is to incorporate morphological and orthographic information. This is achieved by enriching ARSENTB with the in-context morphological features that were extracted at an earlier stage. These features include the ATB-tokens, stems and lemmas (each with and without diacritics). We also extract roots using lemma-root lookup tables for verbs and nouns [146,147]. Also, elongation markers that were extracted during preprocessing are also used to enrich the treebank. Each of these features is mapped into its corresponding node in the detokenized trees. Figure 6.3 shows a sample of an enriched sentiment tree corresponding to the phrase **فرصتان ذهبتان لتحقيق الفوززززز** *frStAn \*hbytAn ltHqyyq Alfuzzzzzz* ‘two golden opportunities to winnnnnn’.

### 6.3.2 Description and Intrinsic Evaluation

The corpus we used to generate ARSENTB consists of 1,177 comments sampled by [138] from the QALB corpus. It is worth mentioning that this dataset is the same one we used to evaluate RAE for OMA in chapter 5, where we disregarded the neutral cases leaving us with 1,133 comments.

Parsing the 1,177 comments produced a total of 123,242 nodes that correspond to phrases ranging from words (leaves) to full comments (roots). Since we target out-of-context annotations, we do not need multiple annotations of the same phrase, even if it appears in different contexts. Therefore, we compiled a list of

Sentiment Tree				
word: raw	فرصتان <i>frStAn</i>	ذهبيتان <i>*hbytAn</i>	لتحقيق <i>ltHqyq</i>	الفوزرززز <i>Alfwzzzz</i>
word: elongation#	فرصتان	ذهبيتان	لتحقيق	#الفوز
stem: diac	فُرْصَ	ذَهَيْي	تَحْقِيق	فَوْز
lemma: diac	فُرْصَة	ذَهَيْي	تَحْقِيق	فَوْز
root	ف-ر-ص	ذ-ه-ب	ح-ق-ق	ف-و-ز

Figure 6.3: The different morphological features provided in ARSENTB

78,879 unique phrases to be randomly distributed to annotators. Each phrase is independently annotated by 3 to 5 annotators, and annotations are aggregated based on majority voting. To resolve cases with tied votes, we back-off from five to three sentiment classes  $\{negative, neutral, positive\}$  to match annotations of same polarity, regardless of the intensity. If votes remain tied, we assign an additional annotator to break the tie.

Table 6.1 shows the sentiment distribution in ARSENTB across the sets of unique phrases, all phrases and all comments (roots). We also observed that short phrases tend to be neutral as they mostly consist of words that need more context to become expressive, whereas stronger sentiment, particularly negative, builds up in longer phrases.

Figure 6.4 illustrates the normalized sentiment distribution at different levels of  $n$ -grams in ARSENTB. It can be observed that shorter phrases tend to be neutral as they mostly consist of words that need further context to become expressive, whereas stronger sentiment, particularly negative, builds up in longer phrases.

A sentiment composition takes sentiments of two phrases and produces the sentiment of their union. Table 6.2 illustrates the different types of sentiment compositions that are observed in ARSENTB. For each type, it shows the sentiment distribution of its output.

We can observe from Table 6.2 that the output of combining neutral with

Sentiment	unique phrases	all phrases	comments
very negative	4.2%	3%	13.4%
negative	27.4%	17.7%	<b>49.3%</b>
neutral	<b>47.8%</b>	<b>61.3%</b>	3.7%
positive	19.5%	16%	<b>25.5%</b>
very positive	1.1%	2%	8.1%

Table 6.1: Sentiment distribution in ARSENTB

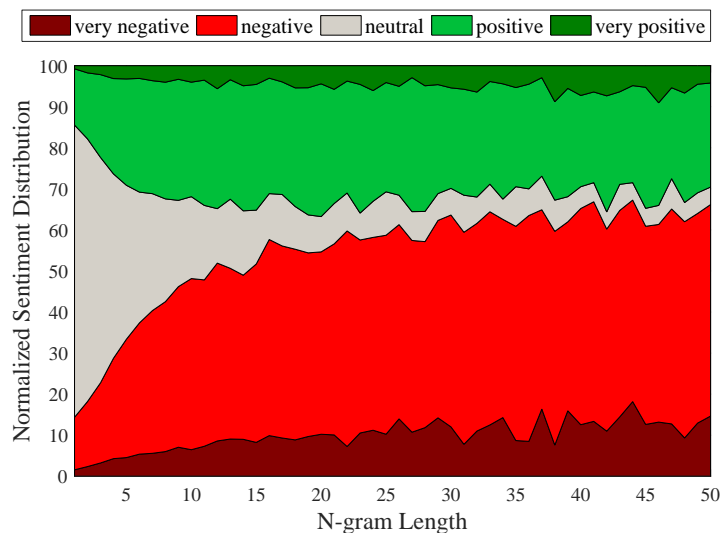


Figure 6.4: Normalized distribution of aggregated sentiment labels at each level of  $n$ -gram in ARSENTB.

subjective phrases tends to be the same as that of the subjective phrase with a probability of 84% for negative and 69% for positive. Also, in more than 90% of the cases, the sentiment remains intact when combining subjective phrases of identical sentiment polarity. However, patterns become less obvious when combining phrases with different sentiments, where the output tends to be negative in 68% of the cases.

We used two methods to evaluate annotation quality. The first method analyzes the ‘per-phrase’ agreement by measuring, for each phrase, how well the annotators agreed on its sentiment. We observed that 47.9% of phrases had full agreement among their annotators, and that 88% had an agreement that is greater than 50%, which allows proper majority aggregation. For the remaining

Type of composition	Sentiment distribution of Output			count
	positive (%)	negative (%)	neutral (%)	
positive + positive	90.3	3.8	5.9	2,874
positive + negative	26.4	68.0	5.6	2,894
positive + neutral	68.9	7.6	23.5	13,385
negative + negative	4.8	93.0	2.2	3,753
negative + neutral	5.6	84.3	10.1	14,460
neutral + neutral	9.7	7.7	82.7	23,488

Table 6.2: Different types of sentiment compositions observed in ARSENTB

12% with tied votes, backing-off from five to three classes resolved most of these cases, and the remaining cases were assigned to an independent annotator to break the tie. These statistics reflect the clarity of the guidelines and resulting annotations.

The second method calculates the inter-annotator agreement (IAA) statistics using a set of 600 phrases that are sampled from the treebank. This set was annotated by the author and was compared to the outcome of aggregating annotations produced by CrowdFlower. 87% of the phrases had the identical sentiment labels, and 90% of the phrases agreed in their sentiment polarities regardless of its intensity. We calculated the kappa statistics to measure the proportion of agreement above what would be expected by chance [148]. Linear kappa was equal to 77%, where labels were assumed equally-important. Weighted kappa was equal to 83%, where labels were assigned different weights based on their frequency. These numbers indicate excellent levels of agreement according to [149].

As a result, the sentiment labels in ARSENTB are considered robust in terms of 1) agreement among CrowdFlower annotators, and 2) agreement between their aggregated labels and the authors’ expectations.

## 6.4 Experiments and Results

In this section, we evaluate the impact of the morphologically-enriched RNTN that we train using ARSENTB for OMA. We highlight the improvements achieved due to the morphological and orthographic features. We also compare to similar improvements achieved on well-known classifiers including SVM and the RAE with the best-performing configuration based on chapter 5.

### 6.4.1 Experimental Setting

The input to RNTN is a set of word embedding vectors that are derived by training the Continuous Bag-of-Words (CBOW) model [64] on all 550,273 comments from QALB. Different embeddings were derived for each morphological feature by training word2vec on different versions of QALB, each reflecting a specific feature. In previous work, elongation was used as explicit features to train machine learning models [91]. For our experiments, we learn embeddings for elongations by training word2vec on QALB with elongated words being normalized then marked.

	Size	Comments #	Phrases #
Train	70%	823	85,622
Dev	10%	118	12,518
Test	20%	235	24,991

Table 6.3: Evaluation splits of ARSENTB

We created the *train*, *dev*, and *test* splits by uniformly sampling from ARSENTB as illustrated in Table 6.3. We used the *train* and the *dev* sets to tune the word embedding size and the learning rate. The model was then evaluated on the *test* set using the parameters that achieved the best performance in the tuning stage. Performance was quantified using accuracy and weighted F1 score, where the weights are determined by the percentage of each class in the *test* set. The model was evaluated at both the phrase and the comment-level. A phrase corresponds to any node in the treebank, whereas a comment corresponds to the root nodes only.

Software and tools that were used to conduct the required experiments are mentioned in Appendix A.

### 6.4.2 Results

To evaluate the impact of morphological and orthographic features on the RNTN sentiment model in Arabic, we compare the baseline RNTN that is trained using raw words to variants of the model trained with the suggested linguistic enrichments.

We compare RNTN to the majority baseline, which automatically assigns the most frequent class in the *train* set to all instances in the *test* set. We also compare to well-known classifiers including SVM and RAE. SVMs are trained using word *n*-grams of different lengths (*n*). Preliminary experiments showed that word *bi*-grams are better than *uni*-grams and *tri*-grams, and hence we report only results

for *bi*-grams. On the other hand, we used the RAE model with the setup that achieved highest results for OMA in chapter 5. In other words, (1) the recursion of RAE is based on syntactic parse trees, (2) the input word vectors are formed by concatenating C&W embeddings with our proposed sentiment embeddings, and (3) the input text is morphologically-tokenized. To ensure a fair comparison with RNTN, we also train SVM and RAE with the same levels of morphological abstraction (word, stem, lemma and root). Table 6.6 illustrates the performances of the different models for five-way and three-way sentiment analysis.

Finally, we conduct a qualitative analysis of the RNTN model in order to evaluate its conformity with the Human Reading for Sentiment (HRS) framework that we introduced in chapter 4.

### Impact of Orthographic features

Experimental results, shown in Tables 6.4 and 6.5 illustrate the performance improvement introduced to the baseline RNTN by marking/normalizing character repetition (elongation) and adding diacritics. The baseline RNTN is trained using raw unprocessed words as they originally appeared in the corpus.

	Phrases		Comments	
	Accuracy	F1-score	Accuracy	F1-score
Baseline RNTN (raw words)	72.4	72.0	52.3	43.4
Normalizing Elongation	73.4	73.2	53.0	46.9
Marking Elongation	73.5	73.4	54.5	48.0
Adding Diacritics	74.5	73.9	54.2	47.5
Marking Elongation & Diacritics	<b>75.0</b>	<b>74.6</b>	<b>55.0</b>	<b>48.0</b>

Table 6.4: The impact of adding orthographic features on the performance of baseline RNTN for five-way classification.

It can be observed that marking elongation did not have much impact at the phrase-level performance, mainly because of its scarce occurrence, as it can be observed in only 0.15% of the words and 0.4% of the phrases in the treebank. However, elongations become more evident at the comment-level, and are identified in 8% of the comments, leading to significant performance improvement. These observations confirm the importance of elongations as sentiment indicators.

It can also be observed that adding diacritics to the words consistently improved the performance at both phrase and comment levels. This observation confirms the value of diacritization at reducing ambiguity.

Based on these observations, the remaining experiments assume that diacritics are known and elongation is marked.



	Phrases		Comments	
	Accuracy	F1-score	Accuracy	F1-score
Baseline (raw words)	75.2	75.1	70.6	68.2
Normalizing Elongation	77.9	77.9	71.5	70.2
Marking Elongation	78.2	77.8	74.7	73.9
Adding Diacritics	78.8	78.0	73.4	73.6
Marking Elongation & Diacritics	<b>79.2</b>	<b>78.2</b>	<b>76.7</b>	<b>74.9</b>

Table 6.5: The impact of adding orthographic features on the performance of baseline RNTN for three-way classification.

## Baselines

First, we compare the baseline RNTN that is trained with raw words (Table 6.6, row 3) to the majority classifier (row 1) and to the basic SVM, also trained with raw words (row 2). Results indicate that the baseline RNTN outperforms the majority classifier. It also outperforms the basic SVM at the phrase-level, but achieves lower results at the comment-level. These results confirm the fact that RNTN does not achieve outstanding performances, as it did in English, when directly applied to Arabic text. This is mostly attributed to the complex morphology of the language.

Next, we explore the value of enriching the RNTN with the morphological features that are provided as part of ARSENTB.

## Impact of Morphological Abstraction

We evaluated RNTN under several morphological abstractions: *words*, *tokens*, *stems*, *lemmas*, and *roots* from most specific to most abstract. ARSENTB contains 20,459 unique words that correspond to 14,262 tokens, 9,842 stems, 8,836 lemmas and 4,669 roots. These numbers reflect the concept hierarchy in Arabic morphology, leading to better generalization as we climb through this hierarchy. However, as mentioned in Section 6.2, increasing the level of abstraction introduces additional lexical ambiguity. For instance, 20% of the unique roots in ARSENTB represent words with different sentiments. This percentage drops to 12% and 11% for lemmas and stems, respectively.

Rows 12-16 show that abstracting away from raw words improves performance, with best results achieved with stems. However, going beyond that level of abstraction and operating at the root-level leads to a performance degradation, reflecting loss of semantic information due to over-generalized representations, especially with roots. This behavior is similar to the classical bias-variance

	Five Classes				Three Classes			
	Phrases		Comments		Phrases		Comments	
	Acc.	F1	Acc.	F1	Acc.	F1	Acc.	F1
1 Majority	60.0	45.8	47.2	31.6	60.0	45.8	63.0	48.5
2 Baseline SVM (raw words)	66.3	59.6	<b>52.8</b>	<b>46.3</b>	68.8	63.4	<b>71.9</b>	<b>69.7</b>
3 Baseline RNTN (raw words)	<b>72.4</b>	<b>72.0</b>	52.3	43.4	<b>75.2</b>	<b>75.1</b>	70.6	68.2
4 SVM (stems)	72.7	68.2	55.3	48.2	75.6	72.7	72.7	71.3
5 SVM (lemmas)	74.2	68.8	56.2	48.6	76.3	73.5	<b>72.7</b>	70.9
6 SVM (roots)	<b>75.0</b>	<b>71.3</b>	<b>56.6</b>	<b>49.0</b>	<b>78.3</b>	<b>76.3</b>	72.6	<b>71.4</b>
7 RAE (words)	–	–	51.0	45.0	–	–	73.7	70.9
8 RAE (ATB tokens)	–	–	51.7	46.2	–	–	75.2	73.0
9 RAE (stems)	–	–	<b>58.0</b>	<b>48.8</b>	–	–	<b>78.0</b>	<b>75.8</b>
10 RAE (lemmas)	–	–	53.6	46.1	–	–	76.9	74.5
11 RAE (roots)	–	–	53.4	45.7	–	–	70.3	69.2
12 RNTN (words)	75.0	74.6	55.0	48.0	79.2	78.2	76.7	74.9
13 RNTN (ATB tokens)	77.2	74.2	55.7	48.2	80.1	78.5	79.2	77.8
14 RNTN (stems)	<b>79.6</b>	78.2	<b>60.0</b>	<b>51.8</b>	<b>83.4</b>	<b>83.1</b>	<b>80.0</b>	<b>79.0</b>
15 RNTN (lemmas)	79.4	<b>78.3</b>	56.6	49.1	83.3	82.9	77.9	75.5
16 RNTN (roots)	77.5	76.1	57.4	48.7	81.0	80.7	72.3	71.2

Table 6.6: Performance of the different RNTN models, and comparison to a variety of SVM classifiers. *Numbers in bold indicate best performance under each section: (1-3), (4-6), (7-11) and (12-16). Numbers with underline indicate best performance across all classifiers.*

dilemma, where we are trying to simultaneously minimize two sources of errors; lexical sparsity and ambiguity. The best tradeoff was achieved at the stem level, achieving absolute average F-score improvements (over the baseline RNTN) of 8% and 10.8% at the phrase and the comment level, respectively, on three-way sentiment classification. Similar improvements are also observed on the five-way classification task. It is worth mentioning that the reported improvements were obtained through one iteration (or cycle) of training/testing. To calculate statistical significance, we repeated the experiments two more times on different training/testing splits. We obtained similar improvements that are statistically significant with 95% confidence at 2 degrees of freedom.

We conducted additional experiments to compare the impact of morphology

abstraction on two well-known classifiers: SVM and RAE. Rows 4-6 indicate that, similar to RNTN, increasing the level of morphological abstraction improves performance compared to the baseline SVM. However, unlike RNTN, the highest performance is achieved with roots; the most abstract representation of text. Such behavior indicates that SVM benefits from morphological abstraction to mitigate the impact of *curse of dimensionality* and *sparsity* associated with BoW features. On the other hand, the semantics of these abstractions are better captured by RNTN. For example, although roots are better than stems and lemmas at reducing lexical sparsity, they are expected to produce worse results because they over-generalize the words’ semantics. The best RNTN model (using stems) outperforms the best SVM model (using roots) by 6.8% and 7.6% absolute average F1-score at the phrase and the comment-level, respectively, on three-way sentiment classification. Similar improvements are also observed on the five-way classification task. It can also be observed that, at the comment-level, improvements due to morphological abstraction are more evident in RNTN than in SVM, suggesting that recursive deep learning models are more able to explore the morphology space to improve sentiment prediction.

Rows 7-11 illustrate the performance of RAE at the comment-level, because RAE was initially proposed for sentence-level classification, with only one softmax classification layer trained on top of each parse tree. Results indicate that the impact of morphology on RAE is similar to that on RNTN, where RAE performs the best at the stem-level, while its accuracy decreases at the root-level due to over-generalization. Also, RNTN performs better than RAE, which is mainly due to phrase-level sentiment prediction that helps modeling sentiment composition and its intricacies. However, this improvement required an expensive sentiment treebank, whereas RAE can be trained on corpora with simple sentence-level annotations.

	Phrases		Comments	
	Accuracy	F1	Accuracy	F1
RNTN (words)	80.7	78.1	45.7	39.7
RNTN (stems)	<b>81.0</b>	<b>78.9</b>	<b>47.1</b>	<b>40.8</b>
RNTN (lemmas)	80.8	78.8	46.1	40.0

Table 6.7: The impact of adding morphological features on RNTN performance in English.

Finally, we evaluated the impact of extending the use of morphological features to recursive deep models in English; a non-MRL. For instance, and on average, a lemma in the English Stanford sentiment treebank covers 1.25 words,

whereas in ARSENTB it covers 2.3 words. To prove this evaluation, we trained RNTN on the Stanford sentiment treebank [34] that is enriched with stems and lemmas extracted using the Stanford CoreNLP toolkit [46]. Results in Table 6.7 indicate that adding morphological features improved RNTN by only 1%, which is much less than what has been achieved in Arabic. Therefore, MRLs would benefit more from incorporating features that reduce their complexity.

### Evaluation against the HRS meta-framework

To address this input, we performed a qualitative analysis of the applicability of the HRS meta-framework to the RNTN and RAE models for OMA, where the aim is to identify human reading-specific gaps in these models. The gap analysis is consistent with what was followed for English in chapter 4 (also published in [24]), where the target model should be able to capture the different aspects of the human reading. This includes the low-level cognitive processes including “*lexical access*”, “*syntactic parsing*” and “*semantic proposition formation*”, and the high-level cognitive processes including “*text model of comprehension*”, “*situation model of interpretation*” and “*inference*” including impact of working memory. The needed automated steps of the HRS model are listed in the header columns of the Table 6.8, which illustrates the qualitative analysis, where the rows show the results of the analysis for RAE and RNTN.

We start by analyzing the RAE-based model for OMA. This model captures very well the low-level processes. It performs “*lexical access*” since it is applied at the word-level. It makes use of syntactic and morphological knowledge to perform morphological segmentation and syntactic parsing (for model recursion). It also incorporates grammatical knowledge as it is trained via word embeddings, capturing distributional syntactic and semantic information of the words. It also captures the “*semantic composition formation*” aspect, as it uses composition functions to derive the meanings of phrases (concepts or ideas), which are later used to derive the sentiment of the text. As for the high-level processes, the RAE model does not capture the “*text model of comprehension*”, as it does not use the purpose of reading to select only relevant phrases or ideas. The “*situation model of interpretation*” aspect reflects the ability to construct notions (i.e., ideas or concepts associated with sentiment) that will be used later for inference. This aspect is captured via our proposed lexicon-based word sentiment embedding, which learns word vectors that can be representative of the word-level sentiment, and that can propagate up the sentence parse tree (via recursion) to capture phrase-level sentiment information (equivalently notions). Finally, RAE captures the “*Inference*” aspect by modeling semantic compositionality, i.e., how does the meanings of words and phrases change when they combine together. Combinations are performed in a recursive manner, following the phrase structure of a sentence, until we derive the overall meaning and sentiment of that sentence. Compositionality is a more realistic approach, that comes closer to

		Low-level Processes		Low-level Processes		
		Syntactic Parsing	Semantic proposition formation	Text model of comprehension	Situation model of interpretation	Inference and Memory
RAE	Available	Syntactic relations are captured via embedding. Also, grammatical structure capture via syntactic parsing, which in turn requires morphological segmentation	Recursively model the meanings of phrases (concepts or ideas) of varying lengths	All text is taken into consideration to infer the overall meaning or sentiment	Notions (text + sentiment) are captured via word sentiment embedding (lexicon embedding)	Inference is modeled through SEMANTIC compositionality
	Gap			No selection of phrases/ideas relevant to the purpose of reading		No account for working memory
RNTN	Available	Syntactic relations are captured via embedding. Also, grammatical structure capture via syntactic parsing, which in turn requires morphological segmentation	Recursively model the meanings of phrases (concepts or ideas) of varying lengths	All text is taken into consideration to infer the overall meaning or sentiment	Word and phrase embeddings are fine-tuned based on sentiment classification error	Inference is modeled through SENTIMENT compositionality (how sentiment changes across all levels of phrase constituency)
	Gap			No selection of phrases/ideas relevant to the purpose of reading		No account for working memory

Table 6.8: A qualitative analysis of the recursive deep models for OMA against aspects of human reading.

the way humans perform inference than training classical machine learning models with  $n$ -gram features and different choices of feature engineering. However, this model does not take into consideration the impact of working memory on inference.

Regarding the RNTN-based model for OMA, this model captures very well the low-level processes, similar to the RAE model, with the use of sophisticated NLP techniques to capture the word morphology and the syntactic structure of the text. As for the high-level processes, similar to RAE, the RNTN does not capture “*text model of comprehension*”. The “*situation model of interpretation*” aspect is captured by fine-tuning all word and phrase representations based on the sentiment classification error while training the model. Hence, each word or phrase will be updated to capture the in-context sentiment, producing accurate notions (text + vector representation of sentiment). Finally, RNTN captures

the “*Inference*” aspect by modeling sentiment compositionality, i.e., how does the meaning and sentiment of words and phrases change when combine together. Combinations are performed in a recursive manner, following the phrase structure of a sentence, until we derive the overall meaning and sentiment of that sentence. Similar to RAE, the RNTN does not take into consideration the impact of working memory on inference.

In summary, and based on this qualitative analysis, both RAE and RNTN models capture almost the same aspects of the human reading but vary in the extent to which they accurately capture those aspects. Overall, RNTN comes closer to the human reading than RAE, which is also reflected in the superior results produced by RNTN. The common gaps in both models are: (1) they do not use the purpose of reading to model the “*text model of comprehension*”, and (2) they do not model the working memory during inference. The first gap can be addressed by performing inference (compositionality) on the subjective portions of the text that are relevant to its sentiment. The second gap can be addressed by incorporating the working memory into inference. An example is the use of LSTM networks that are optimized to keep a recollection of recently-read text (short-term memory) and to refresh this memory by flushing existing information to the long-term memory, which resembles the functionality of the human memory during reading.

It is worth noting that existing deep learning models for OMA can only perform sentence-level classification because recursion is based on sentence parse trees. Therefore, these models need to be extended to become applicable to documents as well. The suggested improvements will be considered for future work.

# Chapter 7

## Conclusion

In this dissertation, we presented a novel meta-framework for opinion mining. This framework was inspired from the humans’ natural process of reading and inferring opinions from text, which involves low-level and high-level processing and that uses background knowledge to infer semantics in general and sentiments in specific. The “human reading for sentiment” (HRS) meta-framework identifies gaps in existing approaches, and introduces further improvements. We showed how to apply HRS to approaches that rely on feature engineering, and others that rely on deep learning. For feature engineering-based methods, we developed new “notions” features to model aspects of the human reading process. We also automated two additional aspects of the human reading; the human working memory that reflects the reader’s short span of attention, and the human’s ability to relate words with close meanings. On the other hand, we modified the neural network architecture in deep learning-based methods. We modeled the notions by producing embedded representations that capture sentiments at multiple levels of the text hierarchy. Experiments showed the performance improvements when applying HRS to both state-of-the-art methods, highlighting its ability to improve methods with already high accuracy.

We also presented new state-of-the-art solutions for opinion mining in Arabic by addressing several Arabic-related challenges with focus on relevance to recursive deep models. We addressed morphological complexity and lexical sparsity and ambiguity by performing composition at a more granular level of morphology, and also by enriching the model with with a variety of morphological and orthographic features. We derived sentiment embeddings to improve the accuracy of the input features with a broader coverage of the words’ semantics. We also used phrase structure parse trees to guide the recursion of composition, and to capture the natural order in which constituents are combined to express meaning in sentences. Furthermore, we introduced ARSENTB, the first Arabic sentiment treebank, to support recursive deep models for opinion mining at different levels of text granularity, starting from the word-level. Experimental results using different datasets indicate that the proposed solutions, applied to both RAE

and RNTN models, achieved significant improvements, in both accuracy and F1, compared to the baseline models.

Arabic text can be found in different forms; it can be written in either Modern Standard Arabic (MSA) or in one of its many dialectal variants. It can contain grammatical mistakes (e.g., incorrect verb-subject agreements) or orthographic misspellings (e.g., typing errors). Furthermore, Arabic can be found in different genres of text with different characteristics. For instance, newswire documents or blogs are usually long carefully written in MSA, microblogs such as tweets are short and contain a lot of noisy data (dialects, misspellings, links, switch coding, etc.), and finally online comments can be short or long, and can be either written in MSA or contain significant amount of noise, depending on the comment’s author.

Our proposed models have various degrees of success with Arabic. For MSA (tested with ATB dataset), our model was able to achieve 87% level of accuracy. For dialectal Arabic and misspellings (tested with Twitter data), our model was able to achieve 77% accuracy. For mixture of MSA and dialects with misspellings (tested with the QALB dataset), our model was able to achieve 80% accuracy. The confidence levels in these results were evaluated by comparing with previous methods and using tests of confidence. They were found to have tests of confidence ranging between 90% and 95%.

Our method works best with Arabic MSA, since the models were derived using tools designed for Arabic MSA. For dialectal Arabic, with grammatical mistakes or misspellings, we trained the models using word embeddings that are learned from a large corpus (QALB) that contains similar phenomena. Hence, the model would be aware of semantic and syntactic relations between MSA and dialectal/incorrect words. The results indicate potentials for more improvements for MSA to reach higher accuracy, and even more potential for dialectal improvements.

Future work would need to develop improved models for sentiment inference, and better methods to handle dialectal Arabic. We aim to extend these recursive deep models to match aspects of human reading for OMA. By qualitatively analyzing these models against the HRS meta-framework, the main aspects that need to be captured are the “*text model of comprehension*” and the working memory. Finally, these models only work at the sentence-level, and hence it is very useful to make them applicable to the document-level as well.



# Appendix A

## Tools and Software

The following tables describes the different tools and software that we used to conduct the different experiments for the purpose of this dissertation.

Processing Steps	Tools and Resources
FRN for feature reduction	FRN Java code provided by authors of [12]
Notions Extraction	Custom script in Java to extract notions and assign working memory weights
Sentiment Classification	LibSVM implementation in Java

Table A.1: Software and tools required to train and evaluate the applicability of the HRS meta-framework to the “FRN” opinion model

---

<sup>1</sup>Embeddings are available in <http://ir.hit.edu.cn/~dytang/>

Processing Steps	Tools and Resources
Semantic Composition and Sentiment Classification	GRNN Java code provided by authors of [13]
Word embedding	(A) Embeddings provided in the original GRNN experiments in [13] <sup>1</sup> . (B) Custom MATLAB script to implement our sentiment embedding. We used SentiWordNet [22] as a source of supervised data.

Table A.2: Software and tools required to train and evaluate the applicability of the HRS meta-framework to the “GRNN” opinion model

Processing Steps	Tools and Resources
Semantic Composition and Sentiment Classification	The RAE MATLAB code was published by the authors of [33] <sup>2</sup>
Word embedding	(A) Custom MATLAB script to implement the Collobert and. Weston (C&W) embedding model [99]. (B) Custom MATLAB script to implement our proposed sentiment embedding. We used ArSenL [42] as a source of supervised data.
Morphological tokenization	MADAMIRA [43] to perform morphological tokenization in Arabic.
Syntactic Parsing	(A) Stanford parser [135] <sup>3</sup> , implemented in Java, to generate the Syntactic parse trees. (B) To transform the parse tree’s grammar to a binary grammar, we used the left-factoring algorithm implemented in Python as part of the Natural Language ToolKit (NLTK) [47] <sup>4</sup>

Table A.3: Software and tools required to train and evaluate the “RAE” model for OMA

Processing Steps	Tools and Resources
Semantic Composition and Sentiment Classification	Stanford CoreNLP [12], implemented in Java, to train and evaluate RNTN [34] <sup>5</sup>
Word embedding	Word2vec to obtain the word embeddings, using the CBOW model [64] <sup>6</sup>
Morphological tokenization	MADAMIRA [43] to extract the different morphological and orthographic features.
Syntactic Parsing	(A) Stanford parser [135], implemented in Java, to generate the Syntactic parse trees. (B) To transform the parse tree’s grammar to a binary grammar, we used the left-factoring algorithm implemented in Python as part of the Natural Language ToolKit (NLTK) [47]

Table A.4: Software and tools required to train and evaluate the “RAE” model for OMA

---

<sup>4</sup>The RAE code is available in <http://www.socher.org/index.php/Main/Semi-SupervisedRecursiveAutoencodersForPredictingSentimentDistributions>

<sup>4</sup>Stanford Parser can be downloaded from <http://nlp.stanford.edu/software/lex-parser.shtml>

<sup>4</sup>Left-factoring code can be downloaded from [http://www.nltk.org/\\_modules/nltk/tree.html](http://www.nltk.org/_modules/nltk/tree.html)

<sup>6</sup>Word2vec can be downloaded from <https://github.com/dav/word2vec>

# Bibliography

- [1] H. Chen and D. Zimbra, “Ai and opinion mining,” *IEEE Intelligent Systems*, vol. 25, no. 3, pp. 74–80, 2010.
- [2] Rawkes, “The moment twitter lost steve jobs,” 2011.
- [3] L. Hoffman, “Reflecting on twitter and its implications for elections and democracy,” 2013.
- [4] B. Liu and L. Zhang, “A survey of opinion mining and sentiment analysis,” in *Mining text data*, pp. 415–463, Springer, 2012.
- [5] B. Pang and L. Lee, “Opinion mining and sentiment analysis,” *Foundations and trends in information retrieval*, vol. 2, no. 1-2, pp. 1–135, 2008.
- [6] X. Ding, B. Liu, and P. S. Yu, “A holistic lexicon-based approach to opinion mining,” in *Proceedings of the 2008 international conference on web search and data mining*, pp. 231–240, ACM, 2008.
- [7] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 168–177, ACM, 2004.
- [8] L. Zhang and B. Liu, “Identifying noun product features that imply opinions,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pp. 575–580, Association for Computational Linguistics, 2011.
- [9] X. Ding and B. Liu, “Resolving object and attribute coreference in opinion mining,” in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 268–276, Association for Computational Linguistics, 2010.
- [10] W. P. Grabe and F. L. Stoller, *Teaching and researching: Reading*. Routledge, 2013.

- [11] R. Hobeica, H. Hajj, and W. El Hajj, “Machine reading for notion-based sentiment mining,” in *Data Mining Workshops (ICDMW), 2011 IEEE 11th International Conference on*, pp. 75–80, IEEE, 2011.
- [12] A. Abbasi, S. France, Z. Zhang, and H. Chen, “Selecting attributes for sentiment classification using feature relation networks,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 23, no. 3, pp. 447–462, 2011.
- [13] D. Tang, B. Qin, and T. Liu, “Document modeling with gated recurrent neural network for sentiment classification,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pp. 1422–1432, 2015.
- [14] B. Pang, L. Lee, and S. Vaithyanathan, “Thumbs up?: sentiment classification using machine learning techniques,” in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 79–86, Association for Computational Linguistics, 2002.
- [15] B. Pang and L. Lee, “A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts,” in *Proceedings of the 42nd annual meeting on Association for Computational Linguistics*, p. 271, Association for Computational Linguistics, 2004.
- [16] P. D. Turney, “Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews,” in *Proceedings of the 40th annual meeting on association for computational linguistics*, pp. 417–424, Association for Computational Linguistics, 2002.
- [17] T. Wilson, J. Wiebe, and R. Hwa, “Just how mad are you? finding strong and weak opinion clauses,” in *aaai*, vol. 4, pp. 761–769, 2004.
- [18] A. Abbasi, H. Chen, and A. Salem, “Sentiment analysis in multiple languages: Feature selection for opinion classification in web forums,” *ACM Transactions on Information Systems (TOIS)*, vol. 26, no. 3, p. 12, 2008.
- [19] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity in phrase-level sentiment analysis,” in *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pp. 347–354, Association for Computational Linguistics, 2005.
- [20] Y. Dang, Y. Zhang, and H. Chen, “A lexicon-enhanced method for sentiment classification: An experiment on online product reviews,” *IEEE Intelligent Systems*, vol. 25, no. 4, pp. 46–53, 2010.
- [21] A. Esuli and F. Sebastiani, “Sentiwordnet: A publicly available lexical resource for opinion mining,” in *Proceedings of LREC*, vol. 6, pp. 417–422, 2006.

- [22] G. A. Miller, “Wordnet: a lexical database for english,” *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.
- [23] G. E. Hinton, S. Osindero, and Y.-W. Teh, “A fast learning algorithm for deep belief nets,” *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
- [24] L. Denoyer and P. Gallinari, “The wikipedia xml corpus,” in *International Workshop of the Initiative for the Evaluation of XML Retrieval*, pp. 12–19, Springer, 2006.
- [25] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A large-scale hierarchical image database,” in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pp. 248–255, IEEE, 2009.
- [26] A. Krizhevsky, I. Sutskever, and G. E. Hinton, “Imagenet classification with deep convolutional neural networks,” in *Advances in neural information processing systems*, pp. 1097–1105, 2012.
- [27] J. Zhang, C. Zong, *et al.*, “Deep neural networks in machine translation: An overview,” *IEEE Intelligent Systems*, vol. 15, 2015.
- [28] A. Owens, P. Isola, J. McDermott, A. Torralba, E. H. Adelson, and W. T. Freeman, “Visually indicated sounds,” *arXiv preprint arXiv:1512.08512*, 2015.
- [29] A. Graves, “Generating sequences with recurrent neural networks,” *arXiv preprint arXiv:1308.0850*, 2013.
- [30] J. Mitchell and M. Lapata, “Composition in distributional models of semantics,” *Cognitive science*, vol. 34, no. 8, pp. 1388–1429, 2010.
- [31] Y. Bengio, H. Schwenk, J.-S. Senécal, F. Morin, and J.-L. Gauvain, “Neural probabilistic language models,” in *Innovations in Machine Learning*, pp. 137–186, Springer, 2006.
- [32] D. Tarasov, “Deep recurrent neural networks for multiple language aspect-based sentiment analysis of user reviews,” in *Proceedings of International Conference of Computational Linguistics and Intellectual Technologies Dialog-2015*, vol. 2, pp. 53–64, 2015.
- [33] R. Socher, J. Pennington, E. H. Huang, A. Y. Ng, and C. D. Manning, “Semi-supervised recursive autoencoders for predicting sentiment distributions,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 151–161, Association for Computational Linguistics, 2011.

- [34] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, and C. Potts, “Recursive deep models for semantic compositionality over a sentiment treebank,” in *Proceedings of the conference on empirical methods in natural language processing (EMNLP)*, vol. 1631, p. 1642, Citeseer, 2013.
- [35] D. Dowty, “Compositionality as an empirical problem,” *Direct compositionality*, no. 14, pp. 23–101, 2007.
- [36] UNESCO, “World arabic language day,” 2014.
- [37] “Internet world stats,” 2016.
- [38] M. Rushdi-Saleh, M. T. Martín-Valdivia, L. A. Ureña-López, and J. M. Perea-Ortega, “Oca: Opinion corpus for arabic,” *Journal of the American Society for Information Science and Technology*, vol. 62, no. 10, pp. 2045–2054, 2011.
- [39] M. Abdul-Mageed and M. T. Diab, “Awatif: A multi-genre corpus for modern standard arabic subjectivity and sentiment analysis.,” in *LREC*, pp. 3907–3914, Citeseer, 2012.
- [40] M. Abdul-Mageed, M. Diab, and S. Kübler, “Samar: Subjectivity and sentiment analysis for arabic social media,” *Computer Speech & Language*, vol. 28, no. 1, pp. 20–37, 2014.
- [41] M. Abdul-Mageed and M. T. Diab, “Sana: A large scale multi-genre, multi-dialect lexicon for arabic subjectivity and sentiment analysis.,” in *LREC*, pp. 1162–1169, 2014.
- [42] G. Badaro, R. Baly, H. Hajj, N. Habash, and W. El-Hajj, “A large scale arabic sentiment lexicon for arabic opinion mining,” *ANLP 2014*, vol. 165, 2014.
- [43] A. Pasha, M. Al-Badrashiny, M. T. Diab, A. El Kholly, R. Eskander, N. Habash, M. Pooleery, O. Rambow, and R. Roth, “Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic.,” in *LREC*, vol. 14, pp. 1094–1101, 2014.
- [44] N. Habash, R. Eskander, and A. Hawwari, “A morphological analyzer for egyptian arabic,” in *Proceedings of the twelfth meeting of the special interest group on computational morphology and phonology*, pp. 1–9, Association for Computational Linguistics, 2012.
- [45] X. Yu, Y. Liu, X. Huang, and A. An, “Mining online reviews for predicting sales performance: A case study in the movie domain,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 4, pp. 720–734, 2012.

- [46] C. D. Manning, M. Surdeanu, J. Bauer, J. Finkel, S. J. Bethard, and D. McClosky, “The Stanford CoreNLP natural language processing toolkit,” in *Association for Computational Linguistics (ACL) System Demonstrations*, pp. 55–60, 2014.
- [47] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python.* ” O’Reilly Media, Inc.”, 2009.
- [48] T. Wilson, J. Wiebe, and P. Hoffmann, “Recognizing contextual polarity in phrase-level sentiment analysis,” in *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pp. 347–354, Association for Computational Linguistics, 2005.
- [49] C. Whitelaw, N. Garg, and S. Argamon, “Using appraisal groups for sentiment analysis,” in *Proceedings of the 14th ACM international conference on Information and knowledge management*, pp. 625–631, ACM, 2005.
- [50] C. Lin, Y. He, R. Everson, and S. Ruger, “Weakly supervised joint sentiment-topic detection from text,” *Knowledge and Data Engineering, IEEE Transactions on*, vol. 24, no. 6, pp. 1134–1145, 2012.
- [51] Y. Lu, M. Castellanos, U. Dayal, and C. Zhai, “Automatic construction of a context-aware sentiment lexicon: an optimization approach,” in *Proceedings of the 20th international conference on World wide web*, pp. 347–356, ACM, 2011.
- [52] 2000.
- [53] S. M. Mohammad, S. Kiritchenko, and X. Zhu, “Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets,” *arXiv preprint arXiv:1308.6242*, 2013.
- [54] S. Kiritchenko, X. Zhu, and S. M. Mohammad, “Sentiment analysis of short informal texts,” *Journal of Artificial Intelligence Research*, vol. 50, pp. 723–762, 2014.
- [55] C. Havasi, R. Speer, and J. Alonso, “Conceptnet 3: a flexible, multilingual semantic network for common sense knowledge,” in *Recent advances in natural language processing*, pp. 27–29, Citeseer, 2007.
- [56] C. Whissell, “The dictionary of affect in language,” *Emotion: Theory, research, and experience*, vol. 4, no. 113-131, p. 94, 1989.
- [57] A. Agarwal, F. Biadys, and K. R. Mckeown, “Contextual phrase-level polarity analysis using lexical affect scoring and syntactic n-grams,” in *Proceedings of the 12th Conference of the European Chapter of the Association*



- for *Computational Linguistics*, pp. 24–32, Association for Computational Linguistics, 2009.
- [58] Z. Tu, Y. He, J. Foster, J. van Genabith, Q. Liu, and S. Lin, “Identifying high-impact sub-structures for convolution kernels in document-level sentiment classification,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pp. 338–343, Association for Computational Linguistics, 2012.
- [59] I. M. Katakis, I. Varlamis, and G. Tsatsaronis, “Pythia: Employing lexical and semantic features for sentiment analysis,” in *Machine Learning and Knowledge Discovery in Databases*, pp. 448–451, Springer, 2014.
- [60] D. Tang, F. Wei, B. Qin, T. Liu, and M. Zhou, “Cooooolll: A deep learning system for twitter sentiment classification,” *SemEval 2014*, p. 208, 2014.
- [61] Y. Miura, S. Sakaki, K. Hattori, and T. Ohkuma, “Teamx: A sentiment analyzer with enhanced lexicon mapping and weighting scheme for unbalanced data,” *SemEval 2014*, p. 628, 2014.
- [62] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, “Indexing by latent semantic analysis,” *Journal of the American society for information science*, vol. 41, no. 6, p. 391, 1990.
- [63] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent dirichlet allocation,” *Journal of machine Learning research*, vol. 3, no. Jan, pp. 993–1022, 2003.
- [64] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” in *Advances in neural information processing systems*, pp. 3111–3119, 2013.
- [65] X. Glorot, A. Bordes, and Y. Bengio, “Domain adaptation for large-scale sentiment classification: A deep learning approach,” in *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pp. 513–520, 2011.
- [66] Q. V. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *ICML*, vol. 14, pp. 1188–1196, 2014.
- [67] X. Zhang, J. Zhao, and Y. LeCun, “Character-level convolutional networks for text classification,” in *Advances in Neural Information Processing Systems*, pp. 649–657, 2015.
- [68] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, “A convolutional neural network for modelling sentences,” *arXiv preprint arXiv:1404.2188*, 2014.

- [69] O. Irsoy and C. Cardie, “Deep recursive neural networks for compositionality in language,” in *Advances in Neural Information Processing Systems*, pp. 2096–2104, 2014.
- [70] K. S. Tai, R. Socher, and C. D. Manning, “Improved semantic representations from tree-structured long short-term memory networks,” *arXiv preprint arXiv:1503.00075*, 2015.
- [71] M. A. Aly and A. F. Atiya, “Labr: A large scale arabic book reviews dataset.,” in *ACL (2)*, pp. 494–498, 2013.
- [72] M. N. Al-Kabi, N. A. Abdulla, and M. Al-Ayyoub, “An analytical study of arabic sentiments: Maktoob case study,” in *Internet Technology and Secured Transactions (ICITST), 2013 8th International Conference for*, pp. 89–94, IEEE, 2013.
- [73] A. Shoukry and A. Rafea, “Sentence-level arabic sentiment analysis,” in *Collaboration Technologies and Systems (CTS), 2012 International Conference on*, pp. 546–550, IEEE, 2012.
- [74] A. Mountassir, H. Benbrahim, and I. Berrada, “A cross-study of sentiment classification on arabic corpora,” in *Research and Development in Intelligent Systems XXIX*, pp. 259–272, Springer, 2012.
- [75] R. M. Elawady, S. Barakat, and M. E. Nora, “Sentiment analyzer for arabic comments,” *International Journal of Information Science and Intelligent System*, vol. 3, no. 4, pp. 73–86, 2014.
- [76] N. Omar, M. Albared, A. Q. Al-Shabi, and T. Al-Moslmi, “Ensemble of classification algorithms for subjectivity and sentiment analysis of arabic customers’ reviews,” *International Journal of Advancements in Computing Technology*, vol. 5, no. 14, p. 77, 2013.
- [77] N. Farra, E. Challita, R. A. Assi, and H. Hajj, “Sentence-level and document-level sentiment mining for arabic texts,” in *Data Mining Workshops (ICDMW), 2010 IEEE International Conference on*, pp. 1114–1119, IEEE, 2010.
- [78] M. Maamouri, A. Bies, T. Buckwalter, and W. Mekki, “The penn arabic treebank: Building a large-scale annotated arabic corpus,” in *NEMLAR conference on Arabic language resources and tools*, vol. 27, pp. 466–467, 2004.
- [79] M. Maamouri, D. Graff, B. Bouziri, S. Krouna, and S. Kulick, “Ldc standard arabic morphological analyzer (sama) v. 3.1,” *LDC Catalog No. LDC2010L01. ISBN*, pp. 1–58563, 2010.

- [80] M. Abdul-Mageed, M. T. Diab, and M. Korayem, “Subjectivity and sentiment analysis of modern standard arabic,” in *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pp. 587–591, Association for Computational Linguistics, 2011.
- [81] E. Refaee and V. Rieser, “Subjectivity and sentiment analysis of arabic twitter feeds with limited resources,” in *Proceedings of the Workshop on Free/Open-Source Arabic Corpora and Corpora Processing Tools*, p. 16, 2014.
- [82] H. S. Ibrahim, S. M. Abdou, and M. Gheith, “Sentiment analysis for modern standard arabic and colloquial,” *arXiv preprint arXiv:1505.03105*, 2015.
- [83] W. Black, S. Elkateb, H. Rodriguez, M. Alkhalifa, P. Vossen, A. Pease, and C. Fellbaum, “Introducing the arabic wordnet project,” in *Proceedings of the third international WordNet conference*, pp. 295–300, Citeseer, 2006.
- [84] R. Eskander and O. Rambow, “Slsa: A sentiment lexicon for standard arabic,” in *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2545–2550, 2015.
- [85] D. R. Heise, *Expressive order: Confirming sentiments in social actions*. Springer Science & Business Media, 2007.
- [86] M. Abdul-Mageed, M. Korayem, and A. YoussefAgha, “yes we can?: Subjectivity annotation and tagging for the health domain,” 2011.
- [87] S. A. Morsy, “Recognizing contextual valence shifters in document-level sentiment classification,” 2011.
- [88] R. M. Duwairi and M. A. Alshboul, “Negation-aware framework for sentiment analysis in arabic reviews,” in *Future Internet of Things and Cloud (FiCloud), 2015 3rd International Conference on*, pp. 731–735, IEEE, 2015.
- [89] G. Badaro, R. Baly, R. Akel, L. Fayad, J. Khairallah, H. Hajj, W. El-Hajj, and K. B. Shaban, “A light lexicon-based mobile application for sentiment mining of arabic tweets,” in *ANLP Workshop 2015*, p. 18, 2015.
- [90] N. Abdulla, S. Mohammed, M. Al-Ayyoub, M. Al-Kabi, *et al.*, “Automatic lexicon construction for arabic sentiment analysis,” in *Future Internet of Things and Cloud (FiCloud), 2014 International Conference on*, pp. 547–552, IEEE, 2014.
- [91] A. Mourad and K. Darwish, “Subjectivity and sentiment analysis of modern standard arabic and arabic microblogs,” in *Proceedings of the 4th workshop*

- on computational approaches to subjectivity, sentiment and social media analysis*, pp. 55–64, 2013.
- [92] E. Refaee and V. Rieser, “Benchmarking machine translated sentiment analysis for arabic tweets,” in *NAACL-HLT 2015 Student Research Workshop (SRW)*, p. 71, 2015.
- [93] M. Salameh, S. M. Mohammad, and S. Kiritchenko, “Sentiment after translation: A case-study on arabic social media posts,” in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 767–777, 2015.
- [94] N. Y. Habash, “Introduction to arabic natural language processing,” *Synthesis Lectures on Human Language Technologies*, vol. 3, no. 1, pp. 1–187, 2010.
- [95] O. Smrz, “Functional arabic morphology,” *Formal system and implementation PhD Thesis, Charles University, Prague, Czech Republic*, 2007.
- [96] A. El Kholy and N. Habash, “Orthographic and morphological processing for english–arabic statistical machine translation,” *Machine Translation*, vol. 26, no. 1-2, pp. 25–45, 2012.
- [97] F. Alotaiby, S. Foda, and I. Alkharashi, “Arabic vs. english: Comparative statistical study,” *Arabian Journal for Science and Engineering*, vol. 39, no. 2, pp. 809–820, 2014.
- [98] A. Shahrour, S. Khalifa, and N. Habash, “Improving arabic diacritization through syntactic analysis,” in *LREC*, 2016.
- [99] R. Collobert and J. Weston, “A unified architecture for natural language processing: Deep neural networks with multitask learning,” in *Proceedings of the 25th international conference on Machine learning*, pp. 160–167, ACM, 2008.
- [100] R. Socher, C. D. Manning, and A. Y. Ng, “Learning continuous phrase representations and syntactic parsing with recursive neural networks,” in *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*, pp. 1–9, 2010.
- [101] R. Socher, C. C. Lin, C. Manning, and A. Y. Ng, “Parsing natural scenes and natural language with recursive neural networks,” in *Proceedings of the 28th international conference on machine learning (ICML-11)*, pp. 129–136, 2011.

- [102] B. Lemaire and G. Denhière, “Incremental construction of an associative network from a corpus,” 2004.
- [103] M. S. Seidenberg and J. L. McClelland, “A distributed, developmental model of word recognition and naming.,” *Psychological review*, vol. 96, no. 4, p. 523, 1989.
- [104] M. W. Harm and M. S. Seidenberg, “Computing the meanings of words in reading: cooperative division of labor between visual and phonological processes.,” *Psychological review*, vol. 111, no. 3, p. 662, 2004.
- [105] M. S. Seidenberg, “Connectionist models of word reading,” *Current directions in psychological science*, vol. 14, no. 5, pp. 238–242, 2005.
- [106] C. Perry, J. C. Ziegler, and M. Zorzi, “Nested incremental modeling in the development of computational theories: the cdp+ model of reading aloud.,” *Psychological review*, vol. 114, no. 2, p. 273, 2007.
- [107] S. Dehaene, L. Cohen, M. Sigman, and F. Vinckier, “The neural code for written words: a proposal,” *Trends in cognitive sciences*, vol. 9, no. 7, pp. 335–341, 2005.
- [108] J. G. Cromley and R. Azevedo, “Testing and refining the direct and inferential mediation model of reading comprehension.,” *Journal of Educational Psychology*, vol. 99, no. 2, p. 311, 2007.
- [109] P. Van den Broek, “Using texts in science education: Cognitive processes and knowledge representation,” *Science*, vol. 328, no. 5977, pp. 453–456, 2010.
- [110] O. Etzioni, M. Banko, and M. J. Cafarella, “Machine reading.,” in *AAAI*, vol. 6, pp. 1517–1519, 2006.
- [111] O. Etzioni, M. Cafarella, D. Downey, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates, “Unsupervised named-entity extraction from the web: An experimental study,” *Artificial intelligence*, vol. 165, no. 1, pp. 91–134, 2005.
- [112] M. Banko, M. J. Cafarella, S. Soderland, M. Broadhead, and O. Etzioni, “Open information extraction for the web,” in *IJCAI*, vol. 7, pp. 2670–2676, 2007.
- [113] F. Wu and D. S. Weld, “Autonomously semantifying wikipedia,” in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, pp. 41–50, ACM, 2007.

- [114] C. Kwok, O. Etzioni, and D. S. Weld, “Scaling question answering to the web,” *ACM Transactions on Information Systems (TOIS)*, vol. 19, no. 3, pp. 242–262, 2001.
- [115] S. Schoenmackers, O. Etzioni, and D. S. Weld, “Scaling textual inference to the web,” in *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 79–88, Association for Computational Linguistics, 2008.
- [116] D. J. Kurland, “Inference: The process,” 2000.
- [117] K. A. Ericsson and W. Kintsch, “Long-term working memory.,” *Psychological review*, vol. 102, no. 2, p. 211, 1995.
- [118] G. Dewar, “Parenting for the science-minded,” 2012.
- [119] A. Baddeley and G. Hitch, “Working memory,” 2010.
- [120] A. Yessenalina, Y. Yue, and C. Cardie, “Multi-level structured models for document-level sentiment classification,” in *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pp. 1046–1056, Association for Computational Linguistics, 2010.
- [121] D. Tang, B. Qin, and T. Liu, “Learning semantic representations of users and products for document level sentiment classification,” in *Proc. ACL*, 2015.
- [122] V. F. Hopper, *1001 Pitfalls in English Grammar*. Barron’s Educational Series, 1986.
- [123] E. Riloff, S. Patwardhan, and J. Wiebe, “Feature subsumption for opinion analysis,” in *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pp. 440–448, Association for Computational Linguistics, 2006.
- [124] T. O’Keefe and I. Koprinska, “Feature selection and weighting methods in sentiment analysis,” *ADCS 2009*, p. 67, 2009.
- [125] Y. Yang and J. O. Pedersen, “A comparative study on feature selection in text categorization,” in *ICML*, vol. 97, pp. 412–420, 1997.
- [126] A. Yousefpour, R. Ibrahim, and H. N. Abdull Hamed, “A novel feature reduction method in sentiment analysis,” *International Journal of Innovative Computing*, vol. 4, no. 1, 2014.
- [127] E. Breck, Y. Choi, and C. Cardie, “Identifying expressions of opinion in context.,” in *IJCAI*, vol. 7, pp. 2683–2688, 2007.

- [128] A. Hassan, A. Abbasi, and D. Zeng, “Twitter sentiment analysis: A bootstrap ensemble framework,” in *Social Computing (SocialCom), 2013 International Conference on*, pp. 357–364, IEEE, 2013.
- [129] S.-M. Kim and E. Hovy, “Determining the sentiment of opinions,” in *Proceedings of the 20th international conference on Computational Linguistics*, p. 1367, Association for Computational Linguistics, 2004.
- [130] I. Becker and V. Aharonson, “Last but definitely not least: on the role of the last sentence in automatic polarity-classification,” in *Proceedings of the acL 2010 conference Short Papers*, pp. 331–335, Association for Computational Linguistics, 2010.
- [131] Y. LeCun and Y. Bengio, “Convolutional networks for images, speech, and time series,” *The handbook of brain theory and neural networks*, vol. 3361, no. 10, 1995.
- [132] W. Fletcher, “Kfngram,” *Retrieved July*, vol. 29, p. 2009, 2002.
- [133] K. Toutanova, D. Klein, C. D. Manning, and Y. Singer, “Feature-rich part-of-speech tagging with a cyclic dependency network,” in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pp. 173–180, Association for Computational Linguistics, 2003.
- [134] N. Habash and F. Sadat, “Arabic preprocessing schemes for statistical machine translation,” in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pp. 49–52, Association for Computational Linguistics, 2006.
- [135] S. Green and C. D. Manning, “Better arabic parsing: Baselines, evaluations, and analysis,” in *Proceedings of the 23rd International Conference on Computational Linguistics*, pp. 394–402, Association for Computational Linguistics, 2010.
- [136] N. Chomsky, “On certain formal properties of grammars,” *Information and control*, vol. 2, no. 2, pp. 137–167, 1959.
- [137] E. Refaee and V. Rieser, “An arabic twitter corpus for subjectivity and sentiment analysis,” in *LREC*, pp. 2268–2273, 2014.
- [138] N. Farra, K. McKeown, and N. Habash, “Annotating targets of opinions in arabic using crowdsourcing,” in *ANLP Workshop 2015*, p. 89, 2015.

- [139] B. Mohit, A. Rozovskaya, N. Habash, W. Zaghouani, and O. Obeid, “The first qalb shared task on automatic text correction for arabic,” in *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pp. 39–47, 2014.
- [140] A. Go, R. Bhayani, and L. Huang, “Twitter sentiment classification using distant supervision,” *CS224N Project Report, Stanford*, vol. 1, p. 12, 2009.
- [141] E. Kouloumpis, T. Wilson, and J. D. Moore, “Twitter sentiment analysis: The good the bad and the omg!,” *Icwsn*, vol. 11, pp. 538–541, 2011.
- [142] A. Z. Khan, M. Atique, and V. Thakare, “Combining lexicon-based and learning-based methods for twitter sentiment analysis,” *International Journal of Electronics, Communication and Soft Computing Science & Engineering (IJECSCE)*, p. 89, 2015.
- [143] M. Maamouri, A. Bies, S. Kulick, F. Gaddeche, W. Mekki, S. Krouna, B. Bouziri, and Z. Wajdi, “Arabic treebank: Part 1 v 4.1,” *LDC Catalog No. LDC2010T13. ISBN*, 2010.
- [144] N. Habash and F. Sadat, “Arabic preprocessing schemes for statistical machine translation,” 2006.
- [145] S. M. Mohammad, “A practical guide to sentiment annotation: Challenges and solutions,” in *Proceedings of the Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2016.
- [146] N. Habash and O. Rambow, “Magead: a morphological analyzer and generator for the arabic dialects,” in *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pp. 681–688, Association for Computational Linguistics, 2006.
- [147] M. Altantawy, N. Habash, O. Rambow, and I. Saleh, “Morphological analysis and generation of Arabic nouns: A morphemic functional approach,” in *Proceedings of the International Conference on Language Resources and Evaluation, LREC. Valletta, Malta*, 2010.
- [148] J. Cohen, “A coefficient of agreement for nominal scales. educational and psychosocial measurement, 20, 37-46,” 1960.
- [149] J. L. Fleiss, B. Levin, and M. C. Paik, *Statistical methods for rates and proportions*. John Wiley & Sons, 2013.