

---

# Automatic Adaptation and Creation of Domain-specific Lexical Resources for Opinion Mining

---

Master-Thesis von Hanish Goel (Matriculation No. 2712464)  
February 14, 2017

---



TECHNISCHE  
UNIVERSITÄT  
DARMSTADT



UBIQUITOUS  
KNOWLEDGE  
PROCESSING

---

Automatic Adaptation and Creation of Domain-specific Lexical Resources for Opinion Mining

vorgelegte Master-Thesis von Hanish Goel (Matriculation No. 2712464)

Supervisor: Prof. Dr. Iryna Gurevych

Coordinator: Carsten Schnober, Erik-Lân Do Dinh

Tag der Einreichung:

---

# Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Darmstadt, den February 14, 2017

---

(Hanish Goel)

---

---

## Zusammenfassung

---

Meinungen sind für fast alle menschlichen Aktivitäten von zentraler Bedeutung. Mit zunehmend robusteren Methoden zur maschinellen Sprachverarbeitung (Natural Language Processing, NLP) wird die automatische “Stimmungserkennung” oder Sentimentanalyse für verschiedene Geschäftsbereiche und Anwendungen immer attraktiver. Oft erfordert die Sentimentanalyse ein Sentimentlexikon, also eine Liste von positiven und negativen Wörtern. Vorhandene (generelle) Stimmungslexika sind für spezifische Domänen nicht anwendbar, da Wörter unterschiedliche Stimmungspolaritäten und Orientierungen in verschiedenen Domänen und Sprachen haben können. Somit ist die Identifizierung von domänenspezifischen Sentiment-Wörtern oder -Phrasen eine wesentliche Aufgabe in der Sentimentanalyse. Diese Arbeit konzentriert sich auf die automatische Konstruktion von Stimmungslexika, die für eine gegebene Domäne relevant sind. Die vorgeschlagene Methode basiert auf K-means Clustering und Label Propagation mit Hilfe von Word Embeddings (word2vec). Der Algorithmus extrahiert zuerst eine Liste von Sentiment-tragenden Wörtern aus einem domänenspezifischen Korpus und ordnet dann den extrahierten Wörtern eine positive oder negative Polarität zu. Der Algorithmus wird mit Fokus auf unüberwachtes Lernen und Sprachunabhängigkeit entworfen, so dass er leicht auf neue Sprachen und Domänen erweitert werden kann. Darüber hinaus haben wir auch Experimente mit überwachten Methoden (CRFs und Deep Recurrent Neural Network) für den Vergleich der Ergebnisse durchgeführt. Wir evaluierten die Methoden sowohl extrinsisch durch das IGGSA System, als auch intrinsisch mittels Nutzerbefragungen. Unsere Experimente bestätigen, dass Opinion-Mining-Systeme von domänenspezifischen Sentimentlexika profitieren können.

---

---

## Abstract

---

Opinions are central to almost all human activities. With more robust natural language processing (NLP) methods becoming available, (automatic) sentiment analysis is becoming much more attractive to various business domains and applications. Often sentiment analysis requires a sentiment lexicon, which is a list of positive and negative words. Existing general purpose sentiment lexicons are not applicable for specific domains, because words can have different sentiment polarities and orientation in different domains and languages. Thus, identifying domain specific sentiment words or phrases is a crucial task in sentiment analysis. The work in this thesis focuses on automatically constructing sentiment lexicons that are relevant to a given domain of study. The proposed method is based on K-means clustering and label propagation with word embeddings (word2vec). The algorithm first extracts a list of sentiment bearing words from a domain corpus and then assigns positive or negative polarity to the extracted words. The algorithm is designed with the focus on unsupervised learning and language independence, so that it can be easily extended to new languages and domains. In addition, we have also performed experiments with supervised methods (CRFs and Deep Recurrent Neural Network) for comparing the results. We evaluated the methods both extrinsically and intrinsically by IGSA opinion mining system and user surveys. Our experiments confirm that opinion mining systems can benefit from domain specific sentiment lexicons.

---

---

## Contents

---

<b>List of Tables</b>	<b>6</b>
<b>List of Figures</b>	<b>7</b>
<b>1 Introduction</b>	<b>8</b>
1.1 Motivation . . . . .	9
1.2 Goals of this Thesis . . . . .	9
1.3 Outline . . . . .	10
<b>2 Background</b>	<b>11</b>
2.1 Opinion Mining and Sentiment Analysis . . . . .	11
2.2 Distributed Computing . . . . .	12
2.2.1 Introduction to Hadoop Ecosystem . . . . .	14
2.2.2 Apache Spark . . . . .	16
<b>3 Related Work</b>	<b>18</b>
3.1 Dictionary Based Approach . . . . .	18
3.2 Corpus Based Approach . . . . .	19
3.3 Conclusion . . . . .	21
<b>4 Resources</b>	<b>22</b>
4.1 Datasets . . . . .	22
4.1.1 IGSA Dataset . . . . .	22
4.1.2 GEI-Digital Corpus . . . . .	22
4.1.3 Arab Spring . . . . .	22
4.1.4 Sentiment Labeled Data . . . . .	22
4.2 Pre-trained Word Embeddings . . . . .	23
4.3 Sentiment Lexicons . . . . .	24
4.3.1 MPQA Subjectivity Lexicon . . . . .	24
4.3.2 SentiWS . . . . .	24
<b>5 Preprocessing</b>	<b>25</b>
5.1 Word Embedding (Word2Vec) . . . . .	25
5.2 Stopwords Elimination . . . . .	27
<b>6 Methodology</b>	<b>29</b>
6.1 Sentiment Words Extraction . . . . .	30
6.1.1 Unsupervised Method . . . . .	30
6.2 Polarity Assignment . . . . .	35
6.2.1 Label Propagation Theory . . . . .	35
6.2.2 SENTPROP Framework . . . . .	35
<b>7 Experiments</b>	<b>38</b>
7.1 Evaluation Strategy and Metrics . . . . .	38

---

7.1.1	Evaluation Strategy . . . . .	38
7.1.2	Evaluation Metrics . . . . .	38
7.2	Experiments for Sentiment Words Extraction . . . . .	39
7.2.1	K-means Clustering . . . . .	39
7.2.2	Error Analysis . . . . .	43
7.2.3	Supervised Methods (CRFs and DRNN) . . . . .	44
7.3	Experiments for Polarity Assignment . . . . .	47
7.3.1	Recreating Known Sentiment Lexicons . . . . .	48
7.3.2	Varying the Number of Nearest Neighbors (nn) . . . . .	48
7.3.3	Varying the Parameter Beta( $\beta$ ) . . . . .	49
7.3.4	Varying the Number of Seed Words . . . . .	50
7.3.5	Error Analysis . . . . .	51
7.3.6	Discussion of Results . . . . .	51
7.4	Extrinsic Evaluation: IGGSA Opinion Mining System . . . . .	51
7.4.1	Experiment . . . . .	53
7.4.2	Error Analysis . . . . .	54
7.5	Intrinsic Evaluation: Online User Survey . . . . .	54
<b>8</b>	<b>Conclusion</b>	<b>57</b>
<b>9</b>	<b>Future Work</b>	<b>58</b>
	<b>Bibliography</b>	<b>59</b>

---

## List of Tables

---

4.1	A sentence labeled with subjective expression from MPQA Opinion Corpus. . . . .	23
4.2	The sentence from Table 4.1 with the new labels (senti, non-senti). . . . .	23
4.3	Summary of sentiment labeled data. . . . .	23
7.1	Summary of MPQA and SentiWS lexicons. . . . .	39
7.2	Results of K-means Clustering Method. . . . .	40
7.3	Summary of results of K-means algorithm as a function of $C_{th}$ . . . . .	41
7.4	Summary of results of K-means algorithm with varying number of seed words. . . . .	42
7.5	Summary of results of K-means algorithm as a function of number of clusters. . . . .	42
7.6	Summary of results of K-means algorithm as a function of word embedding dimensions. . .	43
7.7	Example sentence from sentiment labeled dataset. . . . .	44
7.8	Count of instances in each dataset. . . . .	44
7.9	Summary of results of CRF models trained with different combination of features. . . . .	45
7.10	Summary of results of CRF models trained with different dimensions of word embeddings. .	46
7.11	Results of DRNN method with 25 dimensional word embeddings. . . . .	46
7.12	Comparison of both supervised and unsupervised methods. . . . .	47
7.13	English: Positive and Negative Seeds. . . . .	47
7.14	German: Positive and Negative Seeds. . . . .	47
7.15	Performance metrics of recreating lexicons MPQA and SentiWS . . . . .	48
7.16	The summary of performance of SENTPROP with the varying values of nn (Nearest Neighbor) . . . . .	49
7.17	Summary of results of SENTPROP with varying values of beta ( $\beta$ ) . . . . .	50
7.18	The summary of the results of SENTPROP performance with varying the number of seed words. . . . .	50
7.19	Extraction rules for verb, noun, and adjective opinion predicates. . . . .	52
7.20	Results of rule based system using a baseline lexicon: PolArt and SentiWS. . . . .	53
7.21	Results of rule based system using the induced domain specific sentiment lexicon. . . . .	53
7.22	Comparison of F-score (%) of Baseline and Domain Specific Sentiment Lexicon . . . . .	53
7.23	The sentence from the gold labeled data. . . . .	54
7.24	The sentence from system results with the domain speicfic lexicon. . . . .	54
7.25	The sentence from system results with the baseline lexicon. . . . .	54
7.26	Few words from the generated lexicon for ArabSpring. . . . .	55
7.27	Confusion matrix with positive and negative class. . . . .	55
7.28	Confusion matrix with positive, negative, and neutral class. . . . .	55
7.29	Krippendorff's alpha for all the surveys. . . . .	55



---

## List of Figures

---

2.1	Architecture of the standard Hadoop Distributed File System. . . . .	15
2.2	Different modules in Spark Stack. . . . .	16
5.1	Semantic regularities encoded by word vector representations (Mikolov et al., 2013c). . . .	25
5.2	Cosine similarity of European countries for the vector representing Sweden. . . . .	26
5.3	The Skip-gram model architecture that predicts surrounding words given the current word (Mikolov et al., 2013a) . . . . .	27
6.1	Schematic Diagram of Methodology of Inducing Domain Specific Sentiment Lexicons. . . .	29
6.2	The first step of inducing a domain specific sentiment lexicon. . . . .	30
6.3	Similar words tend to stay close to each other in a vector space. These are example clusters.	31
6.4	Using the K-means algorithm to form three clusters in a sample data. This is just an illustration. . . . .	33
6.5	Code snippet of K-means clustering implementation using Spark MLlib. . . . .	34
6.6	The second step of inducing a domain specific sentiment lexicon. . . . .	35
6.7	Visual summary of the label propagation framework (Hamilton et al., 2016). . . . .	37
7.1	Precision and recall trends of K-means algorithm with varying values of threshold $C_{th}$ . . . .	40
7.2	Precision and recall trends of K-means algorithm with varying number of seed words. . . .	41
7.3	Precision and recall trends of K-means algorithm with different numbers of clusters. . . .	42
7.4	Precision and recall trends of CRF model with the varying values of sigma. . . . .	45
7.5	Precision and recall trends of CRF model with the varying values of embedding scale. . . .	46
7.6	Accuracy and Precison trends of SENTPROP performance with the varying number of nearest neighbors (nn). . . . .	48
7.7	Effect of beta ( $\beta$ ) on the SENTPROP performance. . . . .	49
7.8	Effect of number of seed words on the performance of SENTPROP. . . . .	50
7.9	Workflow Pipeline of the Rule-based System (Wiegand et al., 2016). . . . .	52

---

## 1 Introduction

---

Opinions are important to almost all human activities and behaviors. We make our beliefs and choices, to a considerable degree, based upon how others perceive the world. Opinions and related concepts such as emotions, attitudes, sentiments, and evaluations are the field of study of *sentiment analysis*, also called *opinion mining*. The term *opinion mining*, first, appeared in a paper by Dave et al. (2003). He refers the ideal opinion mining system as a tool that “process a set of search results for a given item, generating a list of product attributes (quality, features, etc.) and aggregating opinions about each of them (poor, mixed, good)”. The term *sentiment analysis* appears in papers by Tong (2001), Das and Chen (2001) for automatic analysis of market sentiments and tracking of the predictive judgments. Sentiment analysis focuses on classifying whether the underlying opinion in a given text is positive, negative, or neutral. However, nowadays, the term is seen more broadly for the analysis of opinion, sentiment, and subjectivity in text. Therefore, broadly, “sentiment analysis” and “opinion mining” denote the same area of study. In this thesis, we use the terms, opinion mining and sentiment analysis, interchangeably.

The most basic approach to sentiment analysis is to use a *sentiment lexicon*. A *sentiment lexicon* (or *opinion lexicon*) is a list of words and phrases, each of which is assigned with a positive or negative score reflecting its sentiment polarity and strength. In a sentiment lexicon, the sentiment words, also called opinion words, are commonly used to express positive or negative sentiments. For example, good, wonderful, and amazing are positive sentiment words, and bad, poor, and terrible are negative sentiment words. Sentiment lexicon is instrumental to sentiment analysis as it provides rich sentiment information, and forms the foundation of many sentiment analysis systems (Pang and Lee, 2008; Liu, 2012; Feldman, 2013). Some of the general challenges for compiling sentiment lexicons are.

- Many methods of generating sentiment lexicons require lexical resources (e.g., WordNet). For instance, Hu and Liu (2004) compiled a sentiment lexicon by expanding a set of seed words based on the synonym and antonym structures of the WordNet dictionary. However, these lexical resources may not be available for new languages and domains.
- How do we deal with context dependency of sentiment words?. Sentiment of words are highly determined by context in which they are used. A word can have different meanings and polarities depending on the context.
- Each language has its own grammar, orthography, and style of writing.

With the developing technology and popularity of opinion-rich resources such as blogs, e-news, and the social networking platforms, new opportunities arise to sentiment analysis of specific communities and topics. For example, organizations are using sentiment analysis to examine the tone of 10-K financial reports and investor message boards (Loughran and McDonald, 2011). This also increases the demand of sentiment lexicons generated for specific domains and context. Moreover, the overwhelming volume of data being produced on a daily basis raises the need of a distributed environment that can process text data fastly and efficiently for generating sentiment lexicons.

This work focuses on creating domain specific sentiment lexicons, i.e. the lexicon that contains relevant words from a domain corpus and maps those words to polarities (positive, negative) based on their usage in the domain context, for accurate analysis of opinions and sentiments.

---

## 1.1 Motivation

---

Sentiment lexicons play a crucial role in analyzing subjective properties such as opinions and emotions of text documents. They serve as an indispensable source of features for supervised classifiers (Mohammad et al., 2013) and many sentiment analysis systems (Taboada et al., 2011). However, the sentiment of words is influenced by domain. Many sentiment words have different polarities in different application domains. For example, the word “old” expresses a positive sentiment when we say the antique is old, but it also expresses a negative sentiment when we say the computer is old.

Existing sentiment lexicons (e.g. SentiWS, SentiWordNet) are typically developed for detecting sentiments in domain general text (as in product reviews, customer feedback, etc.). These general-purpose lexicons are not directly applicable for sentiment analysis of specific domain languages (e.g. financial documents, historical texts, or social media forums). The reason is that languages used in these domains differ significantly with regard to style, vocabulary and orthography. In addition, a general purpose sentiment lexicon does not cover all the words that are relevant to a specific domain, because those words do not express any sentiment in a general language. For instance, “Only here, strong natural borders are lacking, hence the large number of fortifications at the French Eastern border (In German: Hier allein fehlen starke natürliche Grenzen, daher die große Anzahl von Festungen an der französischen Ostgrenze)”<sup>1</sup>. The word “natural” in this sentence may express a positive sentiment in historic context. However, the word “natural” does not express any sentiment in a general language and is, thus, not listed in any general-purpose sentiment lexicon. Such generic sentiment lexicons may be inaccurate and are not suitable for new and unrelated domains. They can mislead sentiment analysis by introducing harmful biases and failing to take into account community-specific vernacular or demographic variations in language in use. Therefore, constructing domain specific sentiment lexicons is crucial for opinion mining and sentiment analysis systems.

However, manually constructing domain-specific sentiment lexicons is expensive and requires understanding of linguistic context of the data (as in historical text or financial documents). This motivated us to develop methods for automatic generation of domain specific sentiment lexicons for opinion mining.

---

## 1.2 Goals of this Thesis

---

The thesis aims at automatically generating domain specific sentiment lexicons for opinion mining. The main goal of the thesis to develop methods (Hamilton et al., 2016), which are language-independent and domain-independent, hence applicable for domains such as historic textbooks, financial documents. Unlike traditional approaches that are dependent on lexical resources (e.g. WordNet), our methods are based on only domain specific word embeddings (word2vec) for constructing a lexical graph and propagating sentiment labels. The aim is to identify sentiment bearing words, i.e. the words which are more relevant for sentiment analysis of a given domain, and then assign positive and negative polarities to them. In our methods, a small list of seed words is obtained from sentiment lexicons that exist in German (e.g. SentiWS, (Remus et al., 2010a)) and English (MPQA Lexicon (Wilson et al., 2005a)) language. Expert users can also provide seed words that convey polarity in their specific field, in addition to the existing resources. The generated lexicons are evaluated extrinsically by using them as a basis for an established opinion mining system that make use of sentiment lexicons and can serve as a use case. User surveys provide intrinsic evaluation about the generated lexicons by asking expert users to judge the correctness of automatically assigned word polarities. To achieve this goal, following contributions are made in this work:

---

<sup>1</sup> <http://gei-digital.gei.de/viewer/!image/PPN725781890/38/-/>

- 
- Literature survey of existing corpus based and dictionary based methods for compiling sentiment lexicons, with the focus on unsupervised learning.
  - Design and implementation of an algorithm based on K-means clustering model for extracting sentiment bearing words from a domain corpus.
  - Implementation of an unsupervised algorithm (Hamilton et al., 2016) that combines label propagation approach with word embeddings (Mikolov et al., 2013b) to assign polarities (positive, negative) to words.
  - Scalable solution by implementing the methods in a Spark environment.
  - Intrinsic and extrinsic evaluation of the work by using IGGSA opinion mining system and user surveys. The methods have been tested and validated for German and English language.

---

### 1.3 Outline

---

The thesis has been structured as follows. Chapter 2 discusses the fundamental concepts that are necessary to understand the work. Chapter 3 covers the existing work done in the area of compiling sentiment lexicons. Chapter 4 presents the information about the resources that are used for this work. Chapter 5 discusses the preprocessing steps that are required before performing any experiments. Chapter 6 details on the proposed solution towards inducing a domain specific sentiment lexicon. Chapter 7 presents the experiments and results of evaluation done on our methods. Conclusions and the possible future extensions of this work are presented in Chapter 8 and Chapter 9, respectively.

---

## 2 Background

---

This chapter covers the general background of topics and technologies that is necessary and useful for profound understanding of this thesis. First, an introduction is given about opinion mining and sentiment analysis in general, as the thesis topic is directly connected to it. Then, an overview of distributed computing is presented that includes an introduction to Hadoop and Spark distributed programming paradigm. The methods in this work for inducing domain specific sentiment lexicons have been implemented in a Spark environment for faster and parallel processing of big corpus.

---

### 2.1 Opinion Mining and Sentiment Analysis

---

Opinion mining analyzes opinions that express positive or negative sentiments. We use the following review (Liu, 2012) about a Nikon camera to introduce different elements of opinion mining.

*Posted by: Garry Muller*

*Date: December 20, 2015*

*“(1) I bought a Nikon D5300 camera four months ago. (2) I just love it. (3) The battery life is long (4) The picture quality is awesome.(5) But, my brother thinks it is too heavy for him.”*

1. The review contains a number of opinions, both positive and negative, about Nikon D5300 camera. Sentence (2) has a positive opinion about the Nikon camera as a whole. Sentence (3) conveys a positive opinion about its battery life. Sentence (4) conveys a positive opinion about its picture quality. Sentence (5) conveys a negative opinion on the weight of the Nikon camera. The *target* of the opinion in sentence (2) is Nikon D5300, and in sentence (4) is the picture quality of Nikon D5300.
2. Sentiment words, also called opinion words are the most important indicators of sentiments. The sentiment words can be provided by a sentiment lexicon and are used to determine the overall positive or negative sentiment of an opinion sentence or document. The sentiment word in sentence (2) is love, sentence (4) is awesome, and sentence (5) is heavy. The sentiment of words highly depends on the domain. In sentence (4), the word “heavy” expresses a negative sentiment. However, the word “heavy” also expresses a positive sentiment, for example, in a sentence “the music tone is heavy” from music domain.
3. The *opinion holder* in sentences (2), (3), and (4) is the author of the review (“Garry Muller”), but for sentence (5), it is the brother of the author.
4. The *date* of the review is December 20, 2015. This date is important when one needs to know how opinions change with time and trends.

From these opinions, we can derive the following definition:

**Opinion Definition:** An opinion is a quadruple

$$(g, s, h, t),$$

where  $g$  is the target of the opinion,  $s$  is the sentiment expressed about the target,  $h$  is the holder (source) of the opinion and  $t$  is the time when the opinion was provided.

---

In general, opinion mining is investigated mainly at three levels:

- **Document level:** At this level, the task is to determine whether a whole opinion document indicates a positive or negative sentiment (Pang et al., 2002; Turney, 2002). For instance, given a product review, a system determines whether the overall sentiment of the review is positive or negative about the product. This task is commonly referred to document-level sentiment classification. The assumption of this level of analysis is that the opinion document expresses opinions about a single entity and contains opinions from a single opinion holder. It is not applicable for documents that contain opinions about multiple entities.
- **Sentence level:** The task at this level is to classify sentiments expressed in each sentence. Although, there is no difference between document level and sentence level classification because sentences are short documents, sentence level analysis assumes a sentence contains a single opinion. A document generally has multiple opinions. This level of analysis is closely related to subjectivity classification (Wiebe et al., 1999), which distinguishes objective sentences (sentences that express factual information) from subjective sentences (sentences that express subjective views and opinions).
- **Entity and Aspect level:** Sentiment analysis at both the document level and sentence level do not discover what exactly opinion holders like and dislike. The goal of aspect-based sentiment analysis is to discover the sentiment of every aspect in an opinion document. It is based on the idea that an opinion contains a sentiment (positive or negative) and a target (of opinion). For example, the sentence “Although the service is not that good, I love sushi in this restaurant.” evaluates two aspects, “service” and “sushi”. The sentiment on “sushi” is positive, but the sentiment on “restaurant service” is negative. The service and sushi of an entity “Restaurant” are the opinion targets. Thus, the objective of this level of analysis is to discover sentiments on entities and their aspects. Aspect level analysis can also produce a structured summary of opinions about entities and their aspects.

Sentiment analysis has a wide range of applications in many domains. We will look at some of these applications in the following.

- Individuals and organizations are using social media (e.g., Twitter, blogs, reviews, forum discussions) for decision-making. These days, if one wants to buy a consumer product, one no longer needs to ask friends and family for opinions because public forums on the Web contain many user reviews and discussions about the product. Similarly, organizations do no longer have the need to conduct opinion polls and surveys in order to collect public opinions because there is a wealth of such information available on the web.
- Public opinion polls were linked with Twitter sentiment (O’Connor et al., 2010). People express their opinions in tweets about political candidates. Tumasjan et al. (2010) applied Twitter sentiment to predict election results.
- Twitter data, blogs, and movie reviews can be used to predict box-office revenues for movies (Asur and Huberman, 2010; Joshi et al., 2010; Sadikov et al., 2009).
- Twitter moods can be used to predict the stock market. Bar-Haim et al. (2011) identified expert investors in microblogs and performed sentiment analysis of stocks.

---

## 2.2 Distributed Computing

---

A distributed computing system consists of multiple components which are located on networked computers. Each component in a distributed computing system has its own private memory. They commu-

---

nicate and coordinate their actions by exchanging messages over the communication network. Components that are connected by a network may be spatially separated by any distance. They may be located on separate continents, in the same city, or in the same building.

This definition leads to the following three significant properties of distributed systems (Coulouris et al., 2005).

- **Concurrency:** In a network of computers, programs can be executed concurrently. Resources such as web pages or files can be shared over the network when necessary. The capacity of a system to execute concurrent programs and handle shared resources can be scaled by adding more resources to the network.
- **No global clock:** The coordination of actions between programs often depends on the global notion of the time at which the program's actions happen. However, it is hard for the computers in a network to synchronize their clocks and provide a single global idea of the correct time. This is a consequence of the fact that the only coordination is by passing messages through a network (Coulouris et al., 2005).
- **Independent failures:** Each component in a distributed system can fail independently of others. Network faults isolate the computers that are connected to it and the programs on them are not able to immediately find whether the network has failed or the program somewhere in the system has crashed. This implies a distributed system can continue to function even if some component fails.

The prime motivation for using distributed systems stems from a desire to achieve the following purposes (Jia and Zhou, 2004):

- **Resource sharing:** The resources such as hardware, software and data can be easily shared among users in a distributed system. For example, a group of users can share a printer.
- **Openness:** The openness of distributed systems is determined by whether the system can be extended and reimplemented in various ways. This can be achieved by specifying the software interface of various components and making it available for use by client programs.
- **Concurrency:** Several clients can access a shared resource by sending requests to multiple machines connected by a network at the same time. Services and applications that share resources allow multiple client requests to be processed concurrently.
- **Scalability:** A distributed system can be easily scaled to a large number of machines in order to increase the computing power.
- **Fault-tolerance:** In a distributed system, redundant resources can be installed on multiple machines, so that in the state of software failures or hardware faults, the failures or faults can be detected and tolerated by other machines.
- **Transparency:** Transparency is defined as abstraction from the user and the application programmer of the separation of components in a distributed system, so that the system is seen as a whole rather than as several independent components. Distributed systems can provide many kinds of transparencies such as:
  1. Location transparency, which enables local and remote information to be accessed without knowledge of their physical or network location.
  2. Failure transparency, which conceals failures from users and allow them to complete their tasks despite the failure of software or hardware.



- 
3. Replication transparency, which allows duplicating resources on multiple machines to increase reliability and performance.

In distributed systems, there is a great amount of computational power available. In these days, not only server machines, but also desktop computers are well-equipped with multicore processors. For machine learning methods that involve processing large amounts of data, a distributed system can make use of multicore capabilities efficiently and provide parallelization. Aspects like synchronization and communication, fault-tolerance and security, replication and consistency are difficult to handle within a distributed and heterogeneous environment (Coulouris et al., 2005). However, many distributed computing frameworks such as Hadoop, Spark exist which provide efficient ways to build distributed applications and enable them to scale on multiple machines. This thesis aims to use Hadoop<sup>1</sup> and Spark<sup>2</sup> for providing efficient and scalable methods for inducing domain specific sentiment lexicons for opinion mining.

---

### 2.2.1 Introduction to Hadoop Ecosystem

---

The Apache project Hadoop is an open-source software used for storing, managing, and analyzing a large volume of data, designed to enable applications to work with thousands of processing nodes and petabytes of data. The Hadoop idea was originated by Google in an effort to store and process the large-scale web index on thousands of commodity servers, and has been adopted by many web giants such as Facebook, Yahoo!, Twitter, LinkedIn etc,. Hadoop offers a cost-efficient solution for storing, managing, and analyzing the huge volume of data.

A very basic Hadoop ecosystem includes the distributed computation framework MapReduce (Dean and Ghemawat, 2008) and the distributed storage layer Hadoop Distributed File System (HDFS), which are inspired by Google's MapReduce and GFS publications.

**HDFS:** Three goals are critical for designing a large-scale Big Data storage system: 1) performance, 2) reliability, and 3) total cost of ownership (TCO). The Hadoop Distributed File System (HDFS) <sup>3</sup> is a system that is designed to provide high throughput and fault tolerant storage on commodity hardware. Unlike traditional POSIX distributed file systems (e.g. PVFS, Lustre, Ceph), HDFS is designed to perform write-once-read-many (WORM) type of workloads with streaming access and large data sets (e.g., MapReduce). Standard HDFS employs replication of data blocks on several storage nodes and erasure-coding techniques to protect against hardware failures (Fan et al., 2009).

HDFS builds on the master / slave architecture which consists of three key nodes: NameNode, DataNodes, and Clients, as illustrated in Figure 2.1<sup>4</sup>. The file system namespace is managed by a single *NameNode*, which is responsible for data block placement and DataNode management. Each file in HDFS is divided into blocks of data (usually 64MB in size) and stored between so called DataNodes. The NameNode will decide where to put each block based on a pluggable block placement policy. In standard HDFS, NameNode stores the file system namespace in DRAM for faster access. It also keeps a persistent copy of the file system namespace called *FsImage* on a disk for failures and recovery. Any changes to the file system namespace are recorded in EditLog and periodically merged with the FsImage file for keeping it up-to-date.

---

<sup>1</sup> <http://hadoop.apache.org/>

<sup>2</sup> <http://spark.apache.org/>

<sup>3</sup> Apache.org. Hadoop distributed file system. <http://hadoop.apache.org>.

<sup>4</sup> <https://hadoop.apache.org/docs/r1.2.1/images/hdfsarchitecture.gif>



\_\_\_\_\_



---

---

---

---

---

---

## 2.2.2 Apache Spark

---

Apache Spark (Spark, 2015; Zaharia et al., 2010) is a general-purpose cluster-computing framework, which comes with an ability to run applications 100x faster than Hadoop in memory or 10x faster on disk (Spark, 2015). Spark processes data in-memory on the cluster, which makes repeated access to the same data much faster, while Hadoop MapReduce persists data back to the disk after a map or reduce action. Spark performs better when all the data fits in the memory on a cluster and when tasks require iterative computations that need to pass over the same data many times. It can be viewed as an extension to MapReduce (Dean and Ghemawat, 2008). It supports diverse distributed computations and facilitates re-use of working data sets across multiple parallel operations. The problem of inability of sharing data across multiple maps posed by Hadoop MapReduce is resolved by Spark's potential to keep its data in memory. Apache Spark comes with libraries for graph computation, machine learning, SQL, and streaming. This is depicted in Figure 2.2<sup>5</sup>. It offers an interactive ad hoc querying on large datasets and helps to solve multi-pass iterative algorithms efficiently. Spark facilitates applications to scale up and down elastically on clusters with automatic placement of work based on a data locality.

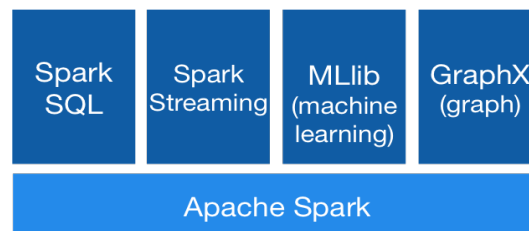


Figure 2.2: Different modules in Spark Stack.


Spark provides programmers with rich Application Programming Interface (APIs) in Scala, Java, and Python. Its integration with the Scala / Python interactive shell eases development and debugging tasks. Spark was initially implemented in Scala. Scala is a general-purpose programming language for a java virtual machine (JVM). It has full support for static type system and type inference. Since source code of Scala is intended to be compiled to Java bytecode and the resulting executable code runs on a java virtual machine, it interoperates easily and can be integrated with other languages on the JVM.

**Resilient Distributed Datasets:** Spark offers unified programming abstractions called Resilient Distributed Datasets (RDD) (Zaharia et al., 2012). They extended the data flow programming model introduced by MapReduce and provided an interface based on coarse-grained transformations on the partitioned data it contains. It allows users to write High-level operators to perform computations without focusing on internals such as work distribution and fault tolerance. They are immutable, and can only be created either through deterministic operations on the data in a stable storage, or transformations on other RDDs. Examples of such transformations are map, filter, and join. RDDs are the primary storage primitives that let users store the data in memory and across multiple compute nodes. They give users direct control of data sharing and make efficient data sharing possible across multiple stages of a parallel computation on a distributed data set. The immutable nature of RDDs simplifies consistency and supports straggler mitigation possible by running back up tasks and backup copies of slow tasks as in MapReduce. Most of the parallel operations performed in real-time are naturally coarse grained. Each operation is applied to many data elements.

Datasets can be stored in Distributed File Systems (DFS), such as the Hadoop Distributed File Systems (HDFS). The data is distributed across the nodes in the HDFS by using either the default or a user defined partitioning scheme. This data can then be loaded as an RDD in Spark. On this data, coarse-grained

---

<sup>5</sup> <http://spark.apache.org/images/spark-stack.png>



---

transformations (map, filter, and join) are performed, which generates a new RDD. On the new RDD, we can either carry out more transformations, or perform an action. An action can return a number, for example “count” returns the number of elements in the dataset, or can store the dataset to the HDFS. A list of all transformations and actions can be found in the Spark programming guide <sup>6</sup>.

---

<sup>6</sup> <https://spark.apache.org/docs/latest/programming-guide.html>

---

### 3 Related Work

---

This chapter presents a literature survey of approaches to compile a sentiment lexicon. There are two main approaches discussed as follows: dictionary-based approach, and corpus-based approach.

---

#### 3.1 Dictionary Based Approach

---

Using a dictionary is an obvious approach to build a sentiment lexicon because most dictionaries like WordNet (Miller et al., 1990) list synonyms and antonyms for each word. The technique of this approach is to use a few sentiment seed words to bootstrap on the basis of the synonym and antonym structure of a dictionary. Practically, this approach works as follows: First, a small list of sentiment words (seeds) with known positive or negative polarity is collected manually, which requires little manual effort. The algorithm then grows this list by searching for their synonyms and antonyms in the WordNet or other online dictionaries. The newly discovered words are added to the seed list. Then, the next iteration begins. The iterative process lasts until no more new sentiment words can be found.

**Kamps et al. (2004)** introduced one of the most sophisticated approaches which focused on a WordNet distance based method to measure the sentiment orientation of a given adjective. They constructed a graph on the adjectives, adding a connection between two adjectives whenever WordNet indicates the presence of a synonymy relation between them. The distance  $d(w1, w2)$  between words  $w1$  and  $w2$  is the length of the shortest path that links  $w1$  and  $w2$  in WordNet. The polarity of an adjective word  $w$  is determined by its relative distance from two seed words “good” and “bad”, i.e.,

$$SO(w) = \frac{d(w, bad) - d(w, good)}{d(good, bad)}. \quad (3.1)$$

$w$  is positive if  $SO(w) > 0$ , and is negative otherwise. The absolute value of  $SO(w)$  indicates the strength of the sentiment.

**Esuli and Sebastiani (2005)** used supervised learning for classifying words into positive and negative categories. In their approach, a set  $P$  of positive seeds terms and a set  $N$  of negative seed terms are first expanded using synonym and antonym relations in lexical dictionaries like WordNet to create the expanded sets  $P'$  and  $N'$ , which represents the training data. The method then finds all the glosses in a dictionary for all the terms in  $P'$  and  $N'$  to construct a feature vector. Using training data and different learning algorithms, a binary classifier is built for predicting the class of words. The process can have many iterations. That is, in each iteration, the newly found positive and negative words with their synonyms and antonyms are added to the training set, then an updated classifier can be built and so on. Esuli and Sebastiani also studied the category objective. They used hyponyms, in addition to synonyms and antonyms, to expand the objective seed set, and then tried various strategies to perform the classification of three classes (Objective, Positive, and Negative). A group of classifiers based on the method of Esuli and Sebastiani (2005) was used to build SentiWordNet, a sentiment lexicon in which each synset of WordNet have two numerical scores PosScore and NegScore, representing how positive and negative the synset is. The objectivity score can be calculated as:  $ObjScore = 1 - (PosScore + NegScore)$ .

In a summary, we note that the advantage of using a dictionary-based approach is that one can quickly and easily discover many sentiment words with their polarity. Though, the resulting list can contain many errors, a manual inspection can be performed to clean it up, which is time-consuming but is only

---

a one-time effort. The main disadvantage is that the sentiment polarity of words determined this way is general or independent of domain and context. In other words, it is difficult to use dictionary based methods to compile domain or context dependent sentiment lexicons. Many words change their sentiment polarity depending on the domain in which they are used. The corpus-based approach discussed below can help deal with this problem.

---

### 3.2 Corpus Based Approach

---

The corpus-based methods use a seed list of known sentiment words (often general-purpose) to discover additional sentiment words and their polarities from a domain corpus. Following, we present some existing works that focus on such methods.

**Hatzivassiloglou and McKeown (1997)** have been the first to propose the idea of using a corpus and a set of seed sentiment adjective words to identify additional sentiment adjectives in a corpus. Their technique explored a set of linguistic rules or conventions on connectives to find more adjective sentiment words and their orientations from a domain corpus. One of the rules is about the conjunction AND, which implies that conjoined adjectives typically have the same sentiment polarity. For instance, in the sentence, “This house is big and wonderful”, if we know the sentiment of big is positive, we can infer that “wonderful” is also having a positive sentiment. The reason is people usually convey the same sentiment on both sides of a conjunction. The following sentence is less likely to be seen, “This house is big and difficult to clean.” It is more acceptable if it is changed to “This house is big, but difficult to clean.”. They also designed rules for other connectives, e.g, OR, EITHER–OR, and NEITHER–NOR (Hatzivassiloglou and McKeown, 1997). Their approach was not always consistent. So, they applied a learning step to decide if two conjoined adjectives have the same or different orientations. First, a graph was constructed with the same and different orientation links between adjectives. Clustering was then executed on the graph to form two sets of words: positive and negative.

Their technique to determine the orientation of adjectives from the conjunctions rules has a following three-step supervised learning algorithm:

1. All conjunctions of adjectives are discovered and extracted from a domain corpus.
2. A set of the extracted conjunctions is divided into a training set and a test set. The conjunctions in the training set are used to train a classifier, which classifies pairs of adjectives either as having the same or as having the different orientation. The classifier is based on a log-linear regression model and is applied to the test set, thus forming a graph with the same or different orientation links among all pairs of adjectives that are conjoined in the test set.
3. A clustering algorithm is run on the graph induced in Step 2 to partition the adjectives into two clusters. By using the intuition that positive adjectives incline to be used more frequently than negative ones, the cluster having the words of higher average frequency in a document set is considered to contain the positive words.

In their experiments (Hatzivassiloglou and McKeown, 1997), the authors used a term set consisting of 657/679 adjectives labeled as being Positive/Negative (it is called the HM term set) and a collection of unlabeled 1987 Wall Street Journal documents.

**Turney and Littman (2003)** introduced a method for inferring the semantic orientation of a given word from its co-occurrence statistics with a set of positive and negative seed words. Basically, it determines the orientation of a word from the strength of its association with a minimal set of positive seed words  $S_p$ , minus the strength of its association with a minimal set of negative seed words  $S_N$ . The pair of two minimal sets of positive and negative words is called a seed set:

- $S_p = \{\text{good, nice, excellent, positive, fortunate, correct, superior}\}$
- $S_n = \{\text{bad, nasty, poor, negative, unfortunate, wrong, inferior}\}$

The strength of semantic association is measured by computing the pointwise mutual information (PMI) of the target word  $w$  with each seed word  $w_i$ . Given a word  $w$ , its orientation value  $O(w)$  (where a positive value implies the positive orientation, and higher value implies the stronger orientation) is given by the following equation

$$O(w) = \sum_{w_i \in S_p} PMI(w, w_i) - \sum_{w_i \in S_n} PMI(w, w_i) \quad (3.2)$$

The authors have evaluated their PMI method on the HM term set from Hatzivassiloglou and McKeown (1997) and also on the General Inquirer lexicon (Stone and Hunt, 1963). The General Inquirer is a text analysis system that defines a large number of categories, each one indicating the presence of a specific trait in a given term. It has two main categories, positive (contains 1,915 terms having a positive polarity) and negative (contains 2,291 terms having a negative polarity). In their experiments, they reduced the list of HM terms to 1,614 (positive) /1,982 (negative) entries (it is called the TL term set) by removing terms appearing in both categories and reducing all the multiple entries of a term in a category with multiple senses to a single entry.

Pointwise mutual information is computed using IR techniques (PMI-IR) and latent semantic analysis (PMI-LSA). The PMI-IR method measures term frequencies and co-occurrence frequencies by querying a set of documents by means of a search engine. The search engine returns a number of matching documents which is used as an estimate of the probabilities needed for PMI. In the experiments, they used AltaVista search engine and three document sets: (i) AV-Eng, consisting of 350 million documents with a total of about 100 billion term occurrences in the English language, indexed by AltaVista at the time of the experiment; (ii) AV-CA, consisting of the AV-Eng 7 million pages with a total of 2 billion term occurrences from .ca domains; and (iii) TASA, consisting of 61,000 documents from Touchstone Applied Science Associates.

The limitation of the methods is that the performance tends to increase with the size of the corpora used, which is quite intuitive because the quality of the co-occurrence data increases with the number of documents on which co-occurrence is computed. The methods need a search engine and heavy computational resources and, therefore, cannot scale well with large corpora. In addition, word co-occurrence only captures “relatedness” of words, but not the similarity of words.

**Wilson et al. (2005a)** focused on contextual subjectivity and sentiments at the phrase or expression level. Contextual sentiment implies that although, a word or phrase in a lexicon is labeled as positive or negative, it may express no sentiment or the opposite sentiment in the context of a sentence. In their approach, first, they labeled the subjective expressions in the corpus, i.e. the expressions that have subjective words or phrases in a given subjectivity lexicon. It is worth to note that a subjectivity lexicon is a bit different from a sentiment lexicon as a subjectivity lexicon may have words that express only subjectivity but no sentiment, e.g., feel, and think. Their goal was to classify the sentiment of the expressions that contain instances of subjectivity clues in the subjectivity lexicon based on the context. They took a supervised learning approach in two steps. The first step detects whether the expression is subjective or objective. In the second step, it determines the polarity of the subjective expression in four categories, positive, negative, both, or neutral, where the category “both” implies there are both positive and negative sentiments. For subjectivity classification, a rich and large set of features were used, which included word features, sentence features, document features, structure features (dependency tree based patterns), and modification features (dependency features). In the second step of sentiment classification, it used features such as word tokens, word sentiments, conj polarity, negations, etc. The both steps employ the machine learning algorithm BoosTexter AdaBoost.HM (Schapire and Singer, 2000) to build classifiers.



---

**Velikovich et al. (2010)** developed a method to compile a sentiment lexicon by using web pages. It was relying on a graph propagation algorithm over a phrase similarity graph. The method assumed as input, a set of positive seed phrases and negative seed phrases. The nodes in the phrase graph represent the candidate phrases chosen from all n-grams up to length 10 collected from 4 billion web pages. They selected only 20 million candidate phrases based on several heuristics such as frequency and mutual information of word boundaries. Then, they constructed a context vector for each candidate phrase using a word window of size six combined over all mentions of the phrase in the 4 billion web pages. The edge set was built through computation of cosine similarity of the context vectors of candidate phrases. All edges  $(v_i, v_j)$  were tossed if they were not one of the 25 highest weighted edges nearest to either node  $v_i$  or  $v_j$ . The weight of edge was set to the corresponding value of cosine similarity. A graph propagation method was used to determine the sentiment of each phrase as the total of all the best paths to the seed set.

**Hamilton et al. (2016)** proposed a SENTPROP framework where they combine domain-specific word embeddings with a label propagation framework to learn accurate domain-specific sentiment lexicons using small sets of seed words. SENTPROP also maintains accurate performance for modestly-sized domain-specific corpora. The framework SENTPROP has two steps: constructing a lexical graph from unlabeled corpora and propagating sentiment labels over this graph. In the first step, lexical graphs are built from distributional word embeddings learned on unlabeled corpora. In the second step, sentiment labels are propagated over the weighted lexical graph using a random walk method (Zhou et al., 2004). Hamilton et al. (2016) showed that their purely corpus-based approach achieved state-of-the-art performance on constructing sentiment lexicons from domain-specific corpora and outperformed methods that rely on hand-curated lexical resources (e.g., WordNet). In their work, they induced and released community-specific sentiment lexicons for 250 online communities from the social media forum Reddit and historical sentiment lexicons for 150 years of English.

---

### 3.3 Conclusion

---

In conclusion, the existing work mostly aims at compiling a sentiment lexicon using hand crafted lexical resources and dictionaries, which are not applicable and available for domains such as historical texts, financial documents, etc. Moreover, sentiment lexicons generated by most of the existing methods can be used for sentiment analysis of general-purpose text such as customer reviews, feedbacks, etc. However, in cases where we want to analyze the sentiment of specific domain languages, it is not enough to simply use methods (Esuli and Sebastiani, 2005; Kamps et al., 2004) to generate sentiment lexicons, because the languages used in specific domains like historic textbooks differ in style and vocabulary. Moreover, sentiment of words differs across domains. For example, the word “soft” is positive in toys domain, but it is negative in the sports domain. Thus, methods for inducing domain-specific sentiment lexicons are crucial to sentiment analysis and opinion mining systems. Although, domain and context dependent sentiment lexicons remain to be extremely challenging even with so much research, recent work by Hamilton et al. (2016) uses domain-specific word embeddings with a label propagation framework to induce accurate domain-specific sentiment lexicons. In this work, the aim is to adopt the idea of Hamilton et al. (2016) to induce domain specific sentiment lexicon for opinion mining.

---

## 4 Resources

---

This chapter presents the resources that we use for our experiments. As part of this thesis, we also constructed a dataset that contains words labeled with two classes, “senti” and “non-senti” for conducting experiments with sentiment words extraction methods (supervised and unsupervised). A detailed description of all the resources used in the work is provided as follows.

---

### 4.1 Datasets

---

---

#### 4.1.1 IGGSA Dataset

---

The IGGSA dataset for German Sentiment Analysis Shared Task 2016 stems from the German-language speeches of the Swiss parliament (Bundesversammlung)<sup>1</sup>. The selected speeches cover many topics such as foreign policies, book price controls, etc. The dataset consists of 1815 sentences with labels of 4935 subjective expression frame instances and 8959 frame element instances (source or target) (Ruppenhofer et al., 2014). It was used in IGGSA shared task 2016 for identifying targets and sources of subjective expressions. We use this dataset for extrinsic evaluation of our methods.

---

#### 4.1.2 GEI-Digital Corpus

---

GEI-Digital Corpus was compiled from historical textbooks between 1850 and 1918<sup>2</sup>, and used for gaining new insights into children and their world. It consists of 600000 pages from a large number of books originated between 1850 and 1918, and was used in a research project Welt der Kinder(WDK)<sup>3</sup>.

---

#### 4.1.3 Arab Spring

---

Arab Spring was compiled from the following news corpora: ZIET, Der\_Spiegel, Die Welt, Frankfurter Rundschau, and taz. We have used this corpus for conducting online user surveys in the context of intrinsic evaluation of our methods. The corpus contains four lac words.

---

#### 4.1.4 Sentiment Labeled Data

---

We require a labeled dataset that contains words annotated with “senti” and “non-senti” labels, for conducting experiments with sentiment words extraction methods (described in Chapter 6). Table 4.2 shows a sample instance annotated with “senti” and “non-senti” labels. The label “senti” means that a word is bearing a sentiment and “non-senti” means that a word does not bear any sentiment. We build the labeled data suitable for our purpose from the MPQA opinion corpus (Wiebe et al., 2005), because it contains many useful annotations such as subjective expressions (SE).

---

<sup>1</sup> The IGGSA dataset is available at [http://iggsasharedtask2016.github.io/data/shata14\\_adjudicated\\_preproc.tgz](http://iggsasharedtask2016.github.io/data/shata14_adjudicated_preproc.tgz)

<sup>2</sup> GEI-Digital, <http://gei-digital.gei.de/viewer/resolver?urn=urn:nbn:de:0220-gd-6230859>

<sup>3</sup> <https://www.ukp.tu-darmstadt.de/research/current-projects/welt-der-kinder/>



For creating this labeled data, we use the MPQA Opinion Corpus version 1.2<sup>4</sup> that contains 535 news articles (11,111 sentences) manually annotated for direct subjective expressions (DSEs) and expressive subjective expressions (ESEs) at the phrase level. These subjective expressions indicate sentiments, emotions, etc. For example, the sentence, shown in Table 4.1, is from the MPQA opinion corpus and has the subjective expression “was apparently simple”, which expresses a positive or negative sentiment. Sentences in MPQA Opinion corpus are labeled using the conventional BIO tagging scheme: B indicates the beginning of a subjective expression, I is used for tokens in the subjective expression, and O indicates tokens outside any subjective expression class.

<b>Sentence</b>	The	issue	at	stake	was	apparently	simple
<b>SE Label</b>	O	O	O	O	B	I	I

Table 4.1: A sentence labeled with subjective expression from MPQA Opinion Corpus.

For converting this MPQA opinion Corpus into our required data format, we annotate context words in subjective expressions of a sentence with “senti” label, ignoring functional words ( e.g. is, has, should, etc.) in the subjective expressions. The remaining words in the sentence are annotated with “non-senti” label. The sentence given in Table 4.1 is, hence, annotated with new labels, as shown in Table 4.2. In the subjective expression, there are two context words “apparently” and “simple”, which are labeled as “senti”, and the functional word “was” in subjective expression and rest of the words in the sentence are labeled as “non-senti”.

<b>Sentence</b>	The	issue	at	stake	was	apparently	simple
<b>Sentiment Label</b>	non-senti	non-senti	non-senti	non-senti	non-senti	senti	senti

Table 4.2: The sentence from Table 4.1 with the new labels (senti, non-senti).

The process is fully automated by a program written in Java environment. We obtained 9232 sentences which contain at least one sentiment word. The dataset is divided into training, development, and test set, as shown in Table 4.3.

<b>Dataset</b>	<b>Number of Instances</b>
Training	6462
Development	924
Test	1846

Table 4.3: Summary of sentiment labeled data.

## 4.2 Pre-trained Word Embeddings

We use word embeddings trained on part of Google News Corpus<sup>5</sup>, which consists of over a 100 billion words, for conducting experiments with data in English. The model contains 300-dimensional vectors for 3 million words and phrases. The phrases were chosen by using a simple data-driven approach described by Mikolov et al. (2013b). For experiments with data in German, we use word embeddings trained by Reimers et al. (2014) for the shared task of GermEval at KONVENS 2014. They trained word2vec word embeddings on 116 Million German sentences extracted from several corpora (e.g. German Wikipedia, the Leipzig Corpora Collection, the print archive of Spiegel, the print archive of ZEIT, etc.). The model contains 100 dimensional word embeddings of 3,363,088 words.

<sup>4</sup> [http://mpqa.cs.pitt.edu/corpora/mpqa\\_corpus/](http://mpqa.cs.pitt.edu/corpora/mpqa_corpus/)

<sup>5</sup> <https://code.google.com/archive/p/word2vec/>

---

## 4.3 Sentiment Lexicons

---

We need a set of positive seed words and a set of negative seed words for our experiments. The seed words are labeled polarity words, i.e. the words for which we already know the sentiment polarity. We use two publicly available sentiment lexicons, SentiWS and MPQA Subjectivity Lexicon, as explained below, for compiling our set of seed words.

---

### 4.3.1 MPQA Subjectivity Lexicon

---

In (Wilson et al., 2005a), MPQA subjectivity lexicon<sup>6</sup> was compiled and made freely available for subjectivity analysis. This lexicon contains a list of 8,222 terms labeled with information about polarity (positive, neutral, and negative), their POS-tagging, and the intensity of the polarity (strong, weak). It has 2,718 positive and 4,911 negative words. The MPQA subjectivity lexicon is domain-independent and does not bear any domain specific information.

---

### 4.3.2 SentiWS

---

SentimentWortschatz, or SentiWS is a publicly available German-language sentiment lexicon (Remus et al., 2010b). It lists negative (e.g., Gefahr = danger) and positive (e.g., Freude = joy) sentiment bearing words weighted within the interval of [-1; 1] plus their part of speech (POS) tag, and their inflections. The current version of SentiWS (v1.8b) consists of 1,650 negative and 1,818 positive words, which add up to 16,406 positive and 16,328 negative word forms, respectively. The SentiWS lexicon was built using Google Translate by translating the English General Inquirer (Stone and Hunt, 1963) as well as few other terms that were mined from positive and negative product reviews.

---

<sup>6</sup> <http://www.cs.pitt.edu/mpqa/>

---

## 5 Preprocessing

---

In this chapter, we describe the preprocessing steps that represent input text documents as vector representations of words using word2vec in a vector space, and after that remove the non-informative features (stop words, special characters, and numbers). We consider these preprocessing steps in detail as follows.

---

### 5.1 Word Embedding (Word2Vec)

---

Mikolov et al. (2013a) proposed the Skip-gram model, an efficient method for learning distributed representations of words that help natural language processing algorithms to achieve better performance by taking the semantic relatedness between words into account during training. For example, until recently, the NLP task sentence prediction made use of n-gram features. If we incorporate distributed word representations as features for sentence prediction during the training phase, the algorithm will lead to a better pattern recognition, because then it will not only look for word co-occurrence patterns, but also for semantic patterns. For instance, the system will be trained to learn that the words “man” and “boy” are semantically more related than the words “man” and “cat”. The system does that based on the statistical inference that the words “man” and “boy” occur more often in similar contexts than the words “man” and “cat”. So, if the system encounters, for instance, the sentence “The man is sick”, and somewhere else it encounters the sentences “The boy is sick” and “The cat is sick”, it will know that the first two sentences are more semantically closer.

The general idea behind word vector models is to use distributional semantics to generate high-dimensional vectors, in which words are represented by context vectors with semantic similarity (Sahlgren, 2005). The representations are called distributed representations because the features are not mutually exclusive and their configurations correspond to the variations seen in the observed data (LeCun et al., 2015). The distributional hypothesis specifies that words with similar meaning have tendency to appear in similar contexts, which indicates that the meaning of words can be derived from its distribution across contexts (Sahlgren, 2005).

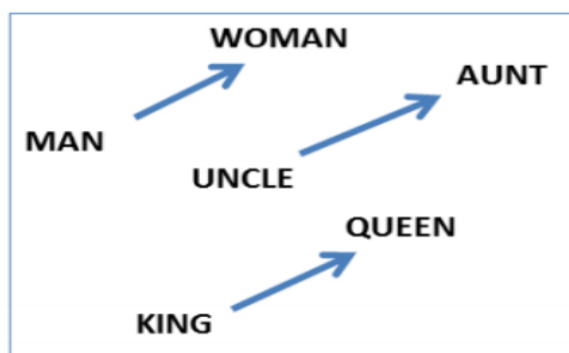


Figure 5.1: Semantic regularities encoded by word vector representations (Mikolov et al., 2013c).

**Similarity encoding:** An interesting fact about these vector spaces is that word vectors can capture various linguistic regularities and relationships of words. Figure 5.1 shows the vector from king to queen is very similar to the vector from man to women (Mikolov et al., 2013c). This vector represents the

high-level concept of genders. Word vectors encode semantic meanings and can have multiple degrees of similarities. In the vector space, shown in Figure 5.1, simple algebraic operations can be performed to exploit the encoded dimensions of similarity. For example, vector (“King”) - vector (“Man”) + vector (“Woman”) results in a vector representation of the word “Queen”.

The word vectors can also be used for finding nearest neighbors of user-specified words and for performing K-means clustering. The cosine distance between two vectors indicates how similar the represented words are to each other. For instance, if we search for nearest neighbors of the word “Sweden”, the distance tool displays Norway, Denmark and Finland as the most similar words. Figure 5.2 shows the names of many European countries found most similar to the word “Sweden”. The similar words also contain the word “Floorball Federation”, which indicates that the word vectors also capture the fact that Floorball was developed in Sweden and is, thus, connected to the word “Sweden”. In conclusion, word vectors can encode many semantic concepts and linguistic properties (e.g. tense, gender, plurality) of words.

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408
floorball_federation	0.529570
luxembourg	0.529477
czech_republic	0.528778
slovakia	0.526340
romania	0.524281
kista	0.522488
helsinki_vantaa	0.519936
swedish	0.519901
balrog_ik	0.514556
portugal	0.502495
russia	0.500196
slovakia_slovenia	0.496051
ukraine	0.495712

Figure 5.2: Cosine similarity of European countries for the vector representing Sweden.

There are many architectures that have been designed to compute such word vectors. In this thesis, we use skip-gram architecture of the word2vec model for computing vector representations. Word2vec was developed by Mikolov et al. (2013a) and provides implementations of two different models for computing word representations: continuous bag-of-words(CBOW) and skip-gram. Word2vec takes a text corpus as input and outputs the word vectors. It first constructs a vocabulary based on all the unique words from the training data, and then learns vector representation of words using similarity matrices. Both the models, CBOW and skip-gram, are described as follows.

**Continuous bag-of-words:** The continuous bag-of-words (CBOW) architecture uses previous and future context to predict the current word. For example, the input words  $w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2}$  are used to output  $w_i$ . Both the history and future words of the current word are taken into account when building a log-linear classifier. In this model, the order of words in the history does not influence the projection.

**Continuous Skip-gram Model:** The continuous skip-gram architecture tends to predict the surrounding words given the current word. Practically, each current word is used as an input to a log-linear model with continuous projection layers, and it predicts context within a certain range before and after the

current word. The more distant words are usually less associated with the current word, so they are given less weight by the model. The formal goal of the Skip-gram model is to maximize the average log probability of a given sequence of words  $w_1, w_2, w_3, \dots, w_T$

$$\frac{1}{T} \sum_{t=1}^T \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t) \quad (5.1)$$

Where  $c$  is the size of the context window and  $w_t$  is the center word. The Skip-gram model formulates  $p(w_{t+j} | w_t)$  using the softmax function:

$$p(w_o | w_i) = \frac{\exp(v_{w_o}'^T v_{w_i})}{\sum_{w=1}^W \exp(v_w'^T v_{w_i})} \quad (5.2)$$

Where  $v_w$  and  $v_{w'}$  are the “input” and “output” vector representations of  $w$ , and  $W$  is the number of words in the vocabulary.

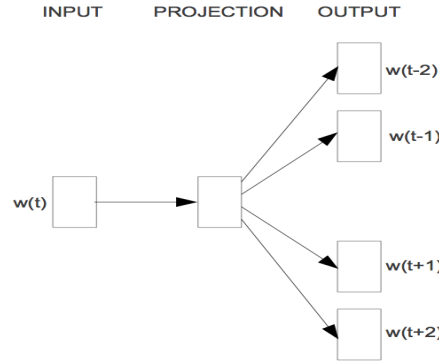


Figure 5.3: The Skip-gram model architecture that predicts surrounding words given the current word (Mikolov et al., 2013a).

We use Deeplearning4j<sup>1</sup> library, which provides the implementation of word2vec skip-gram model, for generating word embeddings on datasets in our work. The process takes the following steps.

1. First, the program loads the raw sentences from a text file, traverses them with an iterator, and performs some sort of preprocessing, such as stripping white space before and after for each line.
2. The second step is tokenization, which is the process of breaking a sequence of text up into words, phrases, or other meaningful elements called tokens. The list of tokens becomes input for Word2vec model. It uses DefaultTokenizerFactory class that creates a new token each time it hits a white space.
3. The third step is training the model. We configure a number of hyperparameters in this step (e.g., minWordFrequency, iterations, layerSize, seed, windowSize).
4. Last, we save the model to a text file.

## 5.2 Stopwords Elimination

The most common words (a, and, also, the, etc.) in any opinion document do not provide any meaning for sentiment analysis. For example, conjunctions such as “or”, “and”, “but” and pronouns “he”, “she”,

<sup>1</sup> <https://deeplearning4j.org/word2vec>

---

“it” etc. These words are treated as stopwords and need to be removed, because they do not have any effect and add any value to sentiment analysis. For the same reason, if a word is a special character or a number, then that word should also be removed. We compiled our list of stopwords for English and German language from online web sources<sup>2</sup>.

---

<sup>2</sup> <http://www.ranks.nl/stopwords>

## 6 Methodology

This chapter describes our work on inducing domain specific sentiment lexicons for opinion mining. We use a modularized approach with *sentence words extraction* and *polarity assignment* (positive or negative) as two independent components. This allows us to experiment with different options for each component.

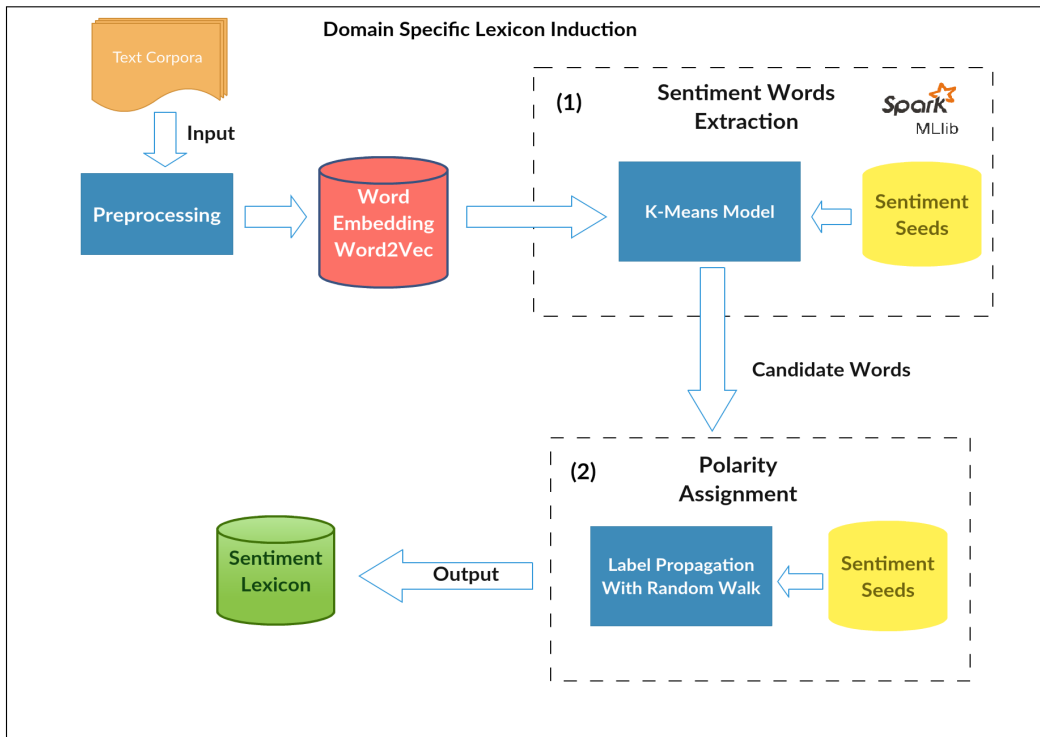


Figure 6.1: Schematic Diagram of Methodology of Inducing Domain Specific Sentiment Lexicons.

After preprocessing of input text documents, that gives us domain specific word embeddings, there are two main steps, as illustrated in Figure 6.1: (1) We extract potential sentiment bearing words based on unsupervised K-means clustering model. This step discovers sentiment bearing words in a domain corpus based on semantic similarity of word clusters with seed words. The input of this step is word embeddings of a given corpus and seed sentiment words. Seed sentiment words are labeled polarity words, for which we already know the sentiment polarity (positive or negative). We define two sets of seed words for positive and negative categories. These seed words can be obtained from available sentiment lexicons such as SentiWS (Remus et al., 2010a), MPQA Subjectivity lexicon (Wilson et al., 2005b). Domain experts can also provide seed words that convey polarity in their specific field. The discovered sentiment words are considered as candidate words for constructing a sentiment lexicon. Candidate words are the words that will be classified by their sentiment polarities in the next step. (2) The second step assigns positive and negative polarity to a list of candidate words that we obtain in the first step. In this step, the input is a list of candidate words and seed sentiment words in the form of word embeddings, and the output is a domain specific sentiment lexicon. This step employs a random walk algorithm to propagate positive and negative labels from known nodes to unknown nodes. The system has been implemented in a Spark

cluster environment for scalability and processing large scale data.

In section 6.1, we discuss the methods we use for the first step *sentiment words extraction*. Then, in section 6.2, we present the *label propagation framework* for labeling the extracted words with positive and negative polarities.

---

## 6.1 Sentiment Words Extraction

---

One of the fundamental challenges of sentiment lexicons is capturing words or expressions for specific domains such as historic textbooks, where opinions are often expressed subtly with regard to style, vocabulary, and orthography. For instance, “Only here, strong natural borders are lacking, hence the large number of fortifications at the French Eastern border (In German: Hier allein fehlen starke natürliche Grenzen, daher die große Anzahl von Festungen an der französischen Ostgrenze)”<sup>1</sup>. The word “natural” in this sentence may express a positive sentiment in historic context. For catching such words, we explore an unsupervised method K-means Clustering with word embeddings (word2vec). Unsupervised learning, having no labeled data, is a task of discovering similar patterns in the data to determine the output. This type of learning does not need any labeled data during the training process. We also tried two supervised methods, conditional random fields (Breck et al., 2007) and deep recurrent neural networks (Irsoy and Cardie, 2014), for data in English. The results of the supervised experiments are provided in chapter 7. Supervised learning requires a training set of sample instances along with the desired label of each of these instances. So, we constructed the labeled data (described in Chapter 4) for experiments with supervised methods. However, constructing a labeled data is not feasible for all domains and languages. Therefore, unsupervised learning seems to be the best choice for languages and domains where labeled data is not available. The unsupervised learning method is described as follows.

---

### 6.1.1 Unsupervised Method

---

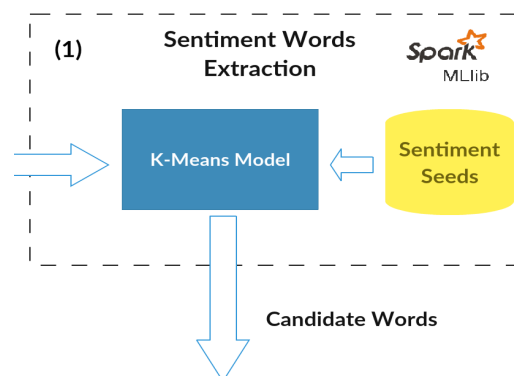


Figure 6.2: The first step of inducing a domain specific sentiment lexicon.

This section presents our work on the first step *sentiment words extraction*, that implements a clustering method with word embeddings. Our idea is to discover sentiment bearing words from a domain corpus based on semantic similarity of clusters of words with a set of seed sentiment words. Clustering, which is an unsupervised learning problem, is a task of partitioning a given data into a certain number of groups whose members are similar in some way. It is based on the notion that words which are semantically close, form the same cluster in a vector space. For example, Figure 6.3 shows that words which have

---

<sup>1</sup> <http://gei-digital.gei.de/viewer/!image/PPN725781890/38/-/>



the same semantic orientation stay close to each other in a vector space. We consider the words of a cluster that contains seed words, as candidate words for constructing a sentiment lexicon. For instance, in Figure 6.3, the cluster 1 contains seed sentiment words “awesome”, “terrific”, and “fantastic”. So, the system will add words of cluster 1 to a list of candidate words. The cluster 3, that does not contain any seed words, will be ignored by the system. We can set a threshold that defines a number of seed words a cluster must have to be considered for the candidate list.

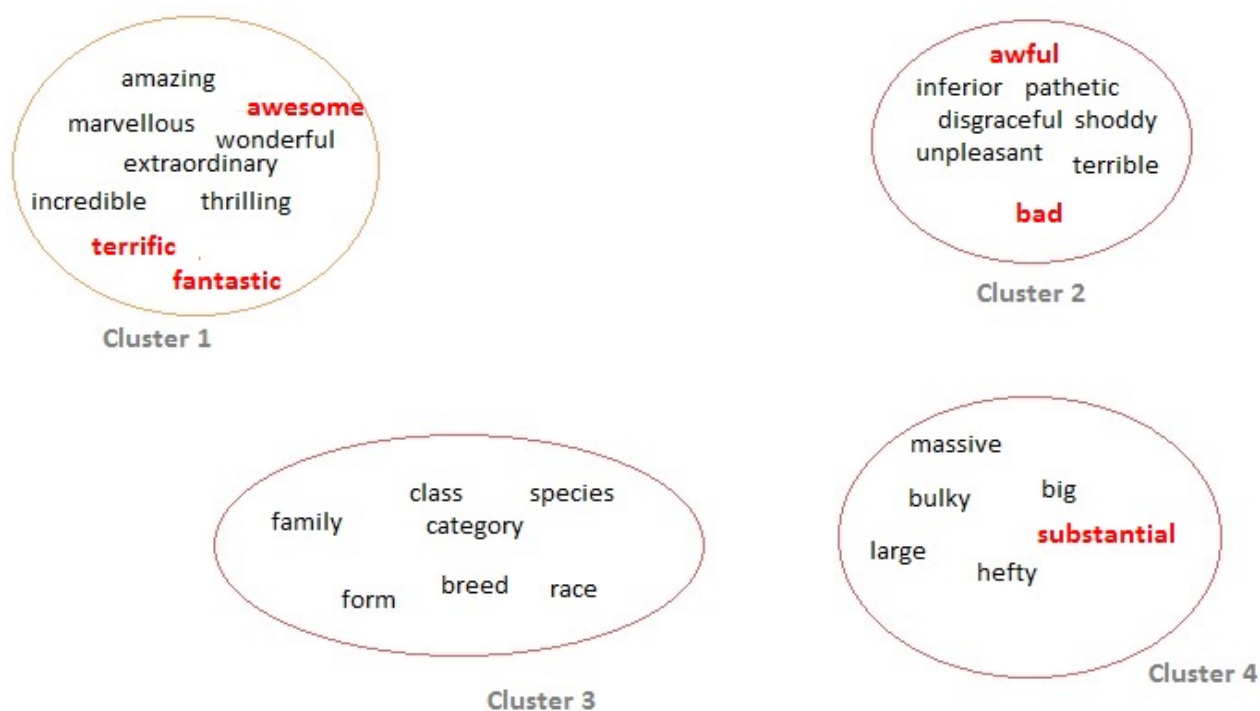


Figure 6.3: Similar words tend to stay close to each other in a vector space. These are example clusters.

We use word2vec word embeddings, which are real-valued vectors and capture many linguistic properties of words (described in Chapter 5), for grouping similar words into a cluster. Since a text corpus can be bigger in size, we need a clustering algorithm that is efficient at run time and computationally fast with a large number of variables (word embeddings can have higher dimensions). Therefore, we adopted the idea of K-means clustering (MacQueen, 1967), which has a time complexity of  $O(n)$ , where  $n$  is the number of data objects.

Our method is implemented using Spark, an open source cluster computing framework, which scales well by simply adding commodity computers (described in Chapter 2). We begin this section with an overview of K-means clustering, and then describe the method to compile a list of candidate words from  $K$  clusters.

---

## K-means Clustering

---

K-means is a center-based clustering technique, which is one of the unsupervised algorithms that solve the well-known clustering problem. The algorithm aims to classify a given data into  $K$  number of groups based on a proximity measure in a vector space. The prototype of K-means is the centroid which is the

mean of a group of points within the dimensional continuous space.

We describe the algorithm as follows. We assume that  $X = x_1, x_2, x_3, \dots, x_n$  be a set of data points, where each point  $x_i$  is a d-dimensional real vector. First, we randomly define  $K$  initial centroids, one for each cluster.  $K$  is the number of clusters desired by the user. The next step is to assign each point in a given data set to the nearest centroid. When all points are assigned, the first iteration is completed. In the next iteration, the center of each cluster resulting from the previous step is recalculated based on the mean of each group and is assigned as a new centroid. After we have these  $K$  new centers, the reassignment of points has to be done to the nearest new center. We repeat this loop and update centroids until no point moves anymore from one cluster to another and each centroid remains the same. K-means is formally described by Algorithm 1.

---

**Algorithm 1** K-Means Clustering

---

- 1: Randomly choose  $K$  points as initial centroid.
  - 2: Calculate the distance between each point and cluster centroids.
  - 3: Assign each point to the nearest cluster center.
  - 4: Recalculate the new cluster center of each cluster.
  - 5: Reassign each data point to the new nearest cluster center.
  - 6: Repeat from step (3) until convergence criterion is met.
- 

We discuss **assignment step**(3) and **update step**(4) of K-means algorithm in more detail.

- Assigning each point to the closest centroid

Euclidean (L2) distance is often used for a proximity measure of data points under consideration. Euclidean distance is a measure of similarity between two vectors in a Euclidean space. However, there are many types of proximity measures that can be used for a given data, such as Manhattan (L1) distance and Hamming distance.

- Centroid and objective function

Step 4 in algorithm aims at minimizing an objective function as Sum of the Squared Error (SSE) function. The objective function expresses the goal of clustering, that is minimizing the squared distance of each data point to the closest centroid. Considering the Euclidean distance for the proximity measure, SSE formula is formally defined as follows:

$$SSE = \sum_{i=1}^K \sum_{x \in c_i} dist(c_i, x)^2 \quad (6.1)$$

Where  $dist$  is the Euclidean (L2) distance between two points in a Euclidean space,  $C_i$  is the  $i^{th}$  cluster, and  $c_i$  is the centroid of cluster  $i$ .

The centroid (mean) of the cluster is defined by:

$$c_i = \frac{1}{m_i} \sum_{x \in c_i} x \quad (6.2)$$

where  $m_i$  is a number of points in  $i^{th}$  cluster

Step 3 in the algorithm directly minimizes the SSE by assigning points to the nearest centroid and step 4 further minimizes the SSE by recomputing the centroid.

Figure 6.4<sup>2</sup> illustrates the steps of K-means algorithm starting with 3 centroids and finding the final clusters in six iterations. The figure shows, for each iteration, the centroid at the beginning of the step and assignment of points to those centroids. In a subsequent step, points are reassigned to updated centroids. In iterations 2 to 6, the centroids move to smaller groups of points and the K-means algorithm terminates, because points do not move anymore.

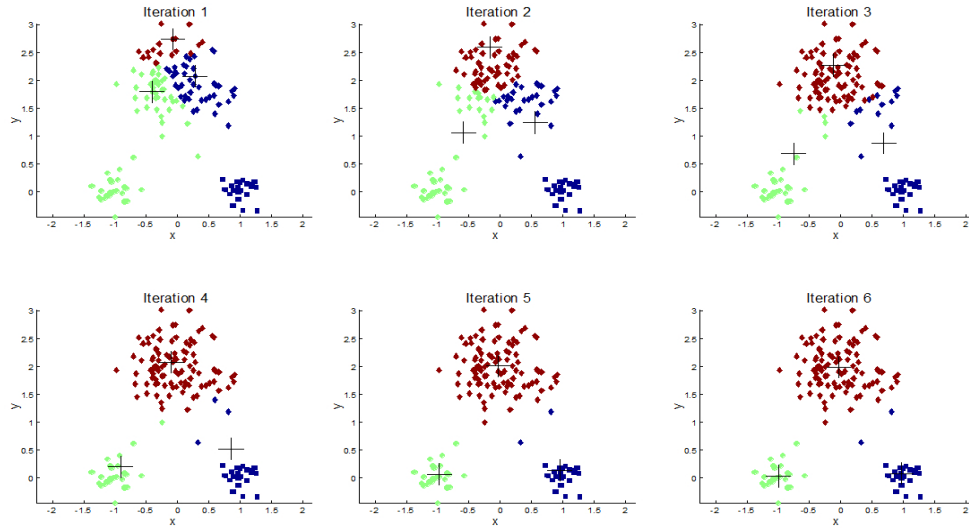


Figure 6.4: Using the K-means algorithm to form three clusters in a sample data. This is just an illustration.

After having clusters of similar words from K-means clustering model, the system compiles a list of candidate words using a set of seed sentiment words.

---

#### Algorithm 2 List of Candidate Words

---

```

1: procedure
2:    $C_{th} \leftarrow$  integer value;
3:    $hitCount \leftarrow 0$ ;
4:    $candidateList \leftarrow null$  # List of candidate clusters;
5:   for Iterate over list of  $K$  clusters do
6:     for iterating over all points in cluster  $C$  do
7:       Match a point with a set of seed sentiment words;
8:       if there is a match then
9:          $hitCount++$ ;
10:    if  $hitCount > C_{th}$  then
11:       $candidateList \leftarrow C$  # adding cluster  $C$  to the list.
12:    else
13:      Ignore cluster  $C$ 
14:  Stores  $candidateList$  in a text file for further steps.

```

---

The approach works as follows: first, the threshold parameter  $C_{th}$  is set to an integer value. The threshold parameter represents the minimum number of seed words that a cluster must have to be considered for a list of candidate words. Second, the system iterates through  $K$  clusters, matching all points in each cluster with a set of seed sentiment words. For a given cluster, if the number of matches found is greater than the threshold value, then the cluster is saved to  $candidateList$ . Once the iteration

<sup>2</sup> <http://jblomo.github.io/datamining290/slides/img/kmeansclustering.jpg>

---

is over, all words belonging to the clusters in *candidateList* will be stored in a text file for further processing in the next component, which is *polarity assignment*. The process is formally defined as in algorithm 2.

---

## Implementation

---

We implement K-means clustering using MLlib library of Spark framework. MLlib<sup>3</sup> is a distributed machine learning library built on Spark. It offers high performance for the iterative ML algorithms with its capacity to store data in memory. It offers ease of use with APIs in Scala, Java, and Python and easy integration with the other components of Spark's ecosystem and the Hadoop workflows. MLlib provides API's for computing basic statistics such as mean, sampling, correlation, that are used as part of many ML algorithms. Some common machine learning algorithms supported by MLlib are clustering, classification, regression, dimensionality reduction, etc.

Figure 6.5 shows the constructs of the K-Means clustering implementation. A text file containing word vectors is passed as an input to the Spark driver program. MLlib uses a parallelized variant of the k-means++ method called `kmeans|`. It accepts the following parameters:

- Number of desired clusters. The effect and optimal value of this parameter are discussed in Chapter 7 (section 7.2).
- Maximum number of iterations to run.
- Number of times to run the K-means algorithm.

```
// Load and parse the data
JavaRDD<String> data = sc.textFile(word2vecFile)

/** Perform clustering*/
JavaPairRDD<Vector, String> parsedData = data.mapToPair(..)

// Cluster the data into five clusters using KMeans
int numClusters = 5
int numIterations = 10

KMeansModel clusters = KMeans.train(parsedData, numClusters, numIterations)
```

Figure 6.5: Code snippet of K-means clustering implementation using Spark MLlib.

The input file is loaded into an Resilient Distributed Datasets (RDD) using the Spark's `textFile()` method, which returns a `JavaRDD` object. The input is parsed to a newer RDD, using a `mapToPair()` transformation on the RDD containing the data points. The new RDD containing the training data is passed to the `train()` method of the `KMeans` object. Once the clustering is performed, we used the `predict()` method to retrieve the clusters, which maps every data point to the cluster index.

---

<sup>3</sup> Spark MLlib: <https://spark.apache.org/mllib/>

---

## 6.2 Polarity Assignment

---

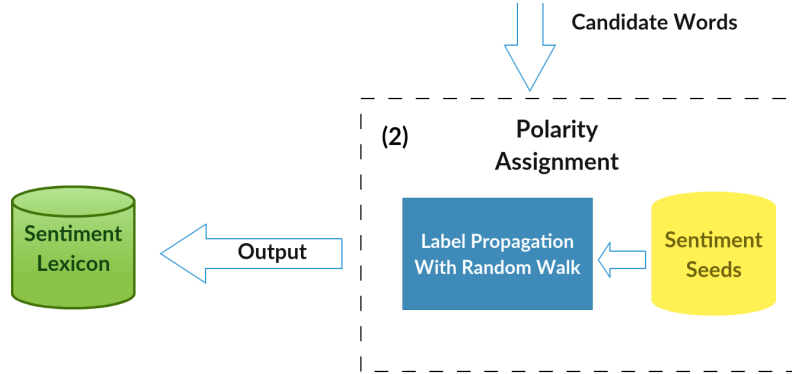


Figure 6.6: The second step of inducing a domain specific sentiment lexicon.

This section focuses on our work to assign positive and negative polarities to a list of candidate words. We implement the approach of SENTPROP framework, which is based on label propagation and word embeddings (Hamilton et al., 2016). The label propagation approach involves propagating the labels from known nodes to unknown nodes (Zhu and Ghahramani, 2002). SENTPROP framework was proposed by Hamilton et al. (2016) to learn accurate sentiment lexicons from unlabeled domain corpora with a small set of seed words. We can see in Figure 6.6, the input of this step is a list of candidate words and a set of seed sentiment words, both in the form of word embeddings, and the output is a sentiment lexicon. First, we describe the label propagation theory, and then, the approach used in the SENTPROP framework.

---

### 6.2.1 Label Propagation Theory

---

Given a graph with labeled and unlabeled nodes, label propagation algorithm propagates the labels recursively through graph edges from the labeled to unlabeled nodes, until a stable result is reached (Zhu and Ghahramani, 2002). Let  $(x_1, y_1) \cdots (x_n, y_n)$  be a set of items and their corresponding labels, where  $X = x_1 \cdots x_n$  are the items and  $Y = y_1 \cdots y_n$  are the item labels. The items  $L = (x_1 \cdots x_l)$  in a dataset have known labels  $Y_L = y_1 \cdots y_l$  and items  $U = (x_{l+1} \cdots x_n)$  have unknown labels  $Y_u = y_{l+1} \cdots y_n$ .  $L$  and  $U$  are often used to denote labeled and unlabeled data respectively. The goal is to predict labels for  $U$  from  $L$ . For achieving this goal, a graph is constructed where nodes are all the data points (words in our context) and weights of edges between nodes represent their similarity. Intuitively, similar data points should have similar labels. There are many algorithms proposed for label propagation theory; however, SENTPROP uses the random walk iterative algorithm (Hamilton et al., 2016; Zhou et al., 2004), which is discussed in the next section.

---

### 6.2.2 SENTPROP Framework

---

The primary aim of the SENTPROP framework is to accurately induce domain specific sentiment lexicons without hand curated resources (e.g. WordNet). In addition, the approach is not specific to certain domains and languages. The input of SENTPROP is divided into two steps: constructing a lexical graph from unlabeled corpora and propagating sentiment labels over this graph. Both the steps are discussed in the following.

---

## Step 1: Building a Lexical Graph

---

The first step is constructing a lexical graph from word embeddings learned on unlabeled corpora. The vector representations of words are computed using skip-gram model described in Chapter 5. After word embeddings are generated, we construct a lexical graph where each word is associated to its  $k$  nearest neighbors according to cosine similarity. The weights of the edges are defined as

$$E_{i,j} = \arccos\left(-\frac{w_i^T w_j}{\|w_i\| \|w_j\|}\right) \quad (6.3)$$

---

## Step 2: Propagating Sentiment Labels over Lexical Graph

---

Given a lexical graph  $G$ , the idea is to propagate sentiment labels from labeled nodes to unlabeled nodes in the graph. The algorithm 3 was proposed and proved to converge by Zhou et al. (2004). In this algorithm, starting with labeled nodes  $(x_1, y_1) \cdots (x_l, y_l)$  labeled with (1 or -1) to unlabeled nodes  $(x_{l+1}, y_{l+1}) \cdots (x_n, y_n)$ , each node propagates its label to its neighbors, and the process is repeated until numeric convergence. At each step, a node gets a contribution from its  $k$  neighbors (weighted by the normalized weight of the edge  $(i, j)$ ) and an additional little contribution from its initial value. In general, the convergence rate of these algorithms can be expected to be at worst on the order of  $O(kn^2)$ , where  $k$  is the number of neighbors of a data point in a lexical graph.

---

**Algorithm 3** Label Propagation, Hamilton et al. (2016); Zhou et al. (2004)

---

- 1: **procedure**
- 2: Input:  $p \in \mathbb{R}^{|V|}$  vector of word-sentiment scores
- 3: Input:  $E$  matrix of edge weights given by equation (3)
- 4: Output:  $P^{-P}(w_i) = \frac{P^P(w_i)}{P^P(w_i) + P^N(w_i)}$
- 5: Compute a transition matrix:  $T = D^{\frac{1}{2}} E D^{\frac{1}{2}}$
- 6: Choose a weighting parameter  $\beta$
- 7: **while**  $p$  has not converged to numeric value **do**

$$p_{(t+1)} = \beta T p_{(t)} + (1 - \beta) s \quad (6.4)$$

8: **End**

---

We discuss each step of this algorithm in detail:

- In step 2 and 3, we construct a vector  $p \in \mathbb{R}^{|V|}$  of word sentiment scores using a seed set  $S$  (e.g. eight positive words). We initialize  $p$  with  $\frac{1}{v}$  in all entries.  $E$  is the matrix of Edge weights which is computed using equation (6.3).
- In step 5, the weight matrix  $E$  is normalized symmetrically by computing  $T = D^{\frac{1}{2}} E D^{\frac{1}{2}}$ , a symmetric transition matrix from  $E$ . This is essential for convergence of the iteration in step 7.  $D$  is a matrix with its elements equal to the column sums of  $E$  on the diagonal.
- In the last step 7, we iteratively update  $p$  using  $T$  until numerical convergence. During each iteration of this step, each point receives the information from its neighbors (first term), and also keeps its initial information (second term). In equation 7,  $s$  is a vector having values  $\frac{1}{|S|}$  for entries corresponding to the seed set  $S$  and zeros otherwise. The term  $\beta$  controls the extent of the relative

amount of the information from its neighbors and its initial label information. Furthermore, the information is propagated symmetrically since  $T$  is a symmetric matrix.

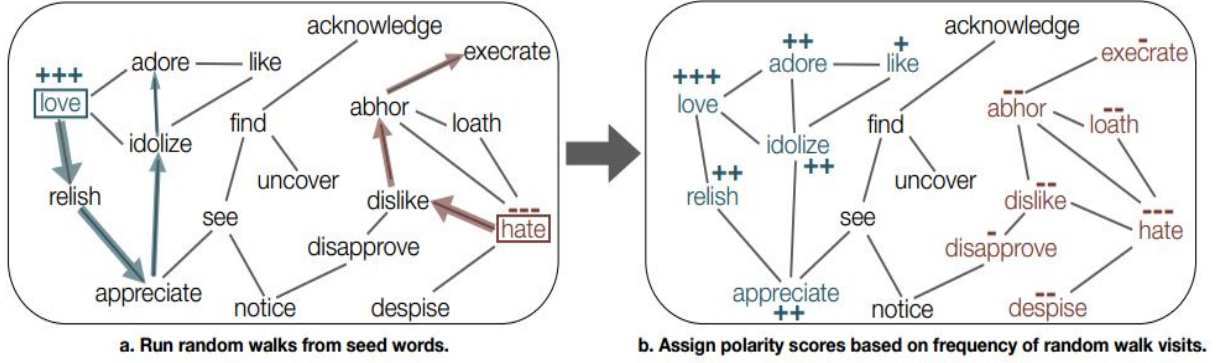


Figure 6.7: Visual summary of the label propagation framework (Hamilton et al., 2016).

The polarity score for a word  $w_i$  is obtained by running random walks using both positive and negative seed sets. As we can see in Figure 6.7, random walks start from seed words to unlabeled nodes, and then each node propagates its label(+ or -) to its neighbors. The polarity score of the nodes will be calculated based on the frequencies of random walk visits from both positive and negative set of seed words. Finally, the label of each unlabeled point is set to be the class of which it has received the most information during the iteration process. As a result, the positive ( $p^P(w_i)$ ) and negative ( $p^N(w_i)$ ) label scores are received, and we combine these values into a positive-polarity score as  $P^P(w_i) = \frac{p^P(w_i)}{p^P(w_i) + p^N(w_i)}$ . The final scores are standardized to have zero mean and unit variance.

For assigning labels to words as positive, neutral, or negative, we use the class-mass normalization method (Zhu et al., 2003). The class-mass normalization (CNN) method estimates the distribution of classes (how many positive, negative, and neutral words there are) from labeled data (labeled data can be any sentiment lexicon) and assigns labels to best match this distribution.



---

## 7 Experiments

---

In this chapter, we present experiments for the previously discussed methods for inducing domain specific sentiment lexicons. This chapter is structured as follows: Section 7.1 presents the evaluation strategy and metrics. Section 7.2 and 7.3 show the experiment results for sentiment words extraction and polarity assignment methods, respectively. Section 7.4 and 7.5 expose the system performance using IGGSA opinion mining system and online user surveys.

---

### 7.1 Evaluation Strategy and Metrics

---

---

#### 7.1.1 Evaluation Strategy

---

The systematic evaluation of an induced sentiment lexicon poses a challenge. The reason is that there are no labeled data that can be used to test and evaluate the correctness of automatically assigned word polarities. There is a necessity to find robust and optimal ways to evaluate the system. We propose the following two methods for evaluation of our system.

---

#### Online User Survey (Human Judgment)

---

In this method, we conducted online user surveys where users were asked to judge the polarity of the words of the induced sentiment lexicon. The details and results of the surveys are presented in section 7.5.

---

#### Evaluating With Opinion Mining System

---

In this method, we evaluated the effectiveness of the induced sentiment lexicon by applying it in IGGSA rule based system for opinion mining. The rule-based system was built for German Sentiment Analysis Shared Task 2014 (Wiegand et al., 2016). The system relies on sentiment lexicons for discovering sources and targets of subjective expressions. We compare the system's performance with our induced sentiment lexicon and general purpose sentiment lexicons (SentiWS and PolArt). The detailed description of the rule-based system and the results of experiments are presented in section 7.4.

---

#### 7.1.2 Evaluation Metrics

---

We use precision, recall, and F-score for performance evaluation of sentiment words extractions, polarity assignment, and the opinion mining system. Precision is what percentage of items that the classifier labeled as positive are actually positive. A high precision means that the majority of items retrieved is relevant. Recall is the number of true positives divided by the total number of items actually belonging to that class. A high recall indicates that the majority of relevant instances is retrieved. The F-score combines precision and recall to give a single score, and is measured as the harmonic mean of precision and recall.

$$Precision = \frac{tp}{tp + fp} \quad (7.1)$$



$$Recall = \frac{tp}{tp + fn} \quad (7.2)$$

$$F1 = \frac{Precision * Recall}{Precision + Recall} \quad (7.3)$$

## 7.2 Experiments for Sentiment Words Extraction

In this section, we present the experiments we conducted for sentiment words extraction methods. First, we discuss the experiments and results of unsupervised method K-means clustering. Then, we discuss the experiment results of two supervised methods, conditional random fields (CRFs) and deep recurrent neural network (DRNN).

### 7.2.1 K-means Clustering

For this experiment, we need a set of seed sentiment words. We obtained the seed words from MPQA Subjectivity Lexicon<sup>1</sup> and SentiWS<sup>2</sup>, which are the most widely used lexicons in the research community. Table 7.1 provides a summary of positive and negative seed words in both the sentiment lexicons.

Lexicon	Positive	Negative
MPQA Subjectivity Lexicon (English)	2,718	4,911
SentiWS (German)	1,818	1,650

Table 7.1: Summary of MPQA and SentiWS lexicons.

First, we conducted experiments with the sentiment labeled data (discussed in Chapter 4) for observing the effect of various factors (e.g. the size of a set of seed words, seed word threshold  $C_{th}$ , etc.) on the performance of the algorithm. Then, we apply the method to Arab Corpus and IGGSA dataset for compiling a list of candidate words for each.

It is noted that the value of the parameter  $K$  (a number of clusters) in K-means clustering is derived from the variable  $P_K$ . The value  $K$  is computed from  $P_K$  percent of the points in a dataset at run-time. For example, if we set the value of  $P_K$  to 5 and total data points (i.e. words) are 3000, then the number of clusters  $K$  will be 150 (5 percent of 3000). In the rest of this section, we use  $P_K$  instead of  $K$  when we refer to a number of clusters.

K-means clustering does not require any training data, so the work is only on the test data. The test set contains 1846 sentences, in which the words are annotated with “senti” and “non-senti” labels (discussed in Chapter 4). Precision and recall are computed by comparing the retrieved sentiment words with the actual sentiment words in the test set. As the data is in English, we used seed words obtained from MPQA lexicon. We ran an experiment with values of  $C_{th}$  set to 5,  $P_K$  set to 7, and obtained the results listed in Table 7.2.

<sup>1</sup> [http://mpqa.cs.pitt.edu/lexicons/subj\\_lexicon/](http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/)

<sup>2</sup> <http://asv.informatik.uni-leipzig.de/download/sentiws.html>

Metrics	Score (%)
Precision	54.61
Recall	81.10
F1	65.26

Table 7.2: Results of K-means Clustering Method.

In the following section, we will look at how the values of parameter  $C_{th}$ , number of clusters, and size of a set of seed words impact the results of K-means clustering method.

#### Effect of Seed Words Threshold $C_{th}$

As discussed in Chapter 6, the parameter  $C_{th}$  represents the minimum number of seed words that a cluster must have to be considered for a list of candidate words. The value we set for  $C_{th}$  is the percentage of data points in a given cluster. In this section, we explore the impact of varying the values of  $C_{th}$  on the performance of K-means algorithm. For these experiments, we set the value of  $P_K$  (number of clusters  $K$  derived from this variable) to 7. As we increase the threshold value, we see that precision soars up and recall declines constantly. The reason is that if a cluster contains a large number of seed sentiment words, it is more likely that the other words in the cluster are sentiment words too, resulting in good precision. But with a high value of threshold  $C_{th}$ , the system would find fewer clusters and recall would be poor. However, F-score sees an overall little change with the increase of the threshold value. Precision and recall are quite sensitive to the choice of threshold. The optimal performance can be achieved when  $C_{th}$  is about 1, although the difference between 1, 3 or 5 is small. Figure 7.1 plots the trends observed with the varying values of  $C_{th}$  in a graph and Table 7.3 shows the performance summary.

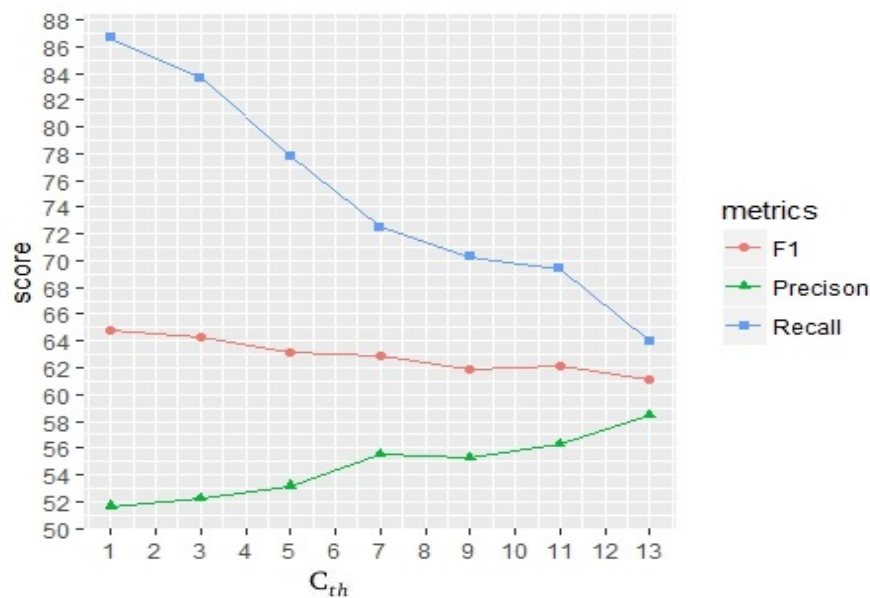


Figure 7.1: Precision and recall trends of K-means algorithm with varying values of threshold  $C_{th}$ .

$C_{th}$	Precision (%)	Recall (%)	F1 (%)
1	51.65	86.68	64.73
5	53.17	77.82	63.17
7	55.56	72.49	62.90
9	55.28	70.29	61.89
11	56.27	69.42	62.16
13	58.42	63.99	61.08

Table 7.3: Summary of results of K-means algorithm as a function of  $C_{th}$ .

### Effect of a number of Seed Words

This section explores the impact of a number of seed sentiment words, as we vary the number from 1000 to 6000 words in an increment of 1000. There is a trade-off between the size of a seed list and the coverage of the generated sentiment lexicon. The larger and better the seed list, the better is the generated lexicon. We can compile a list of seed words from general purpose lexicons such as SentiWS, GI (Stone and Hunt, 1963) or domain experts can provide a list of seed words reflecting polarity in their specific field. Mostly, bigger and better lists of seed words are not either developed or available for some domains and languages. So, here we try to make a trade-off between seed list size and coverage of the generated sentiment lexicon. In Figure 7.2, we see that recall and F-score rise as we increase the number of seed words. We noticed that if the seed list is too small, then the words retrieved are few, and increasing the seed list to a large number results in more words. For example, when we use 1000 seed words, the performance of the algorithm is relatively poor due to their low coverage, although a good precision of 56.86% is achieved. The figure also shows little impact on the F-score when the seed set size exceeds 3000 words, which indicate that the optimal performance can be achieved with a moderate size of a seed set. It should be noted that the performance of the system also depends on the relevance of seed words. The seed words must be relevant to a given domain for finding meaningful sentiment words. Table 7.4 shows the performance summary.

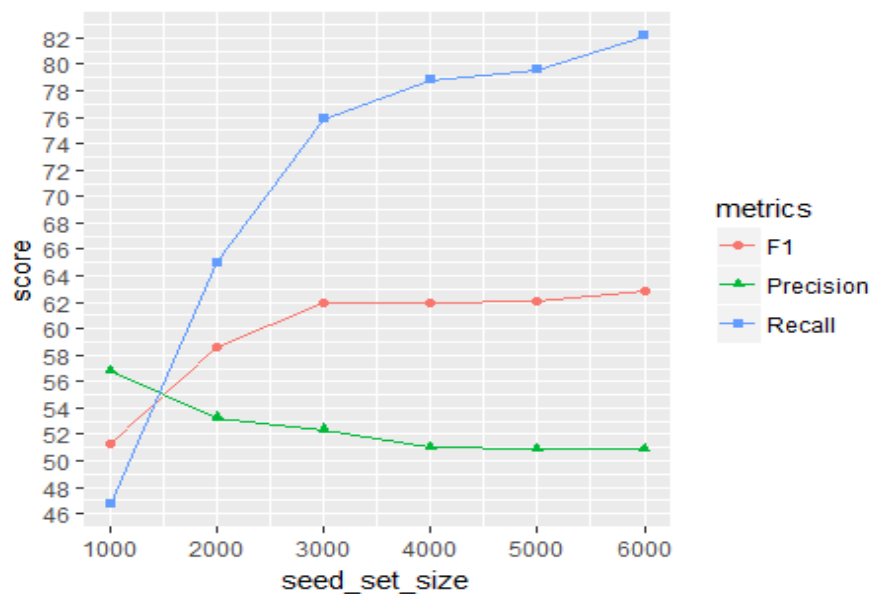


Figure 7.2: Precision and recall trends of K-means algorithm with varying number of seed words.

Seed Set	Precision (%)	Recall (%)	F1 (%)
1000	56.86	46.72	51.29
2000	53.27	65.01	58.56
3000	52.37	75.87	61.97
4000	51.03	78.84	61.96
5000	50.93	79.61	62.12
6000	50.87	82.17	62.84

Table 7.4: Summary of results of K-means algorithm with varying number of seed words.

### Effect of a number of Clusters

As explained in the beginning of this section, the value of  $K$  is derived from variable  $P_K$ . Finding a suitable value of  $K$  is one of the challenges of K-means clustering algorithm. The optimal result for K-means algorithm also depends on the initial cluster centroids. In this section, we present the experiments with different values of  $P_K$  varying from 3 to 15. Figure 7.3 shows that there are very little variations in the values of precision and F-score when we increase the value of  $P_K$ . The reason of this trend is that when we increase the number of clusters, the cluster size reduces, but it does not impact the performance because the parameters such as  $C_{th}$  are computed relatively at run-time based on the size of the cluster. Thus, the system observes less impact of parameter  $K$  on K-means clustering. The best performance can be achieved when the value of  $P_K$  lies between 5 and 9. Table 7.5 shows the performance summary.

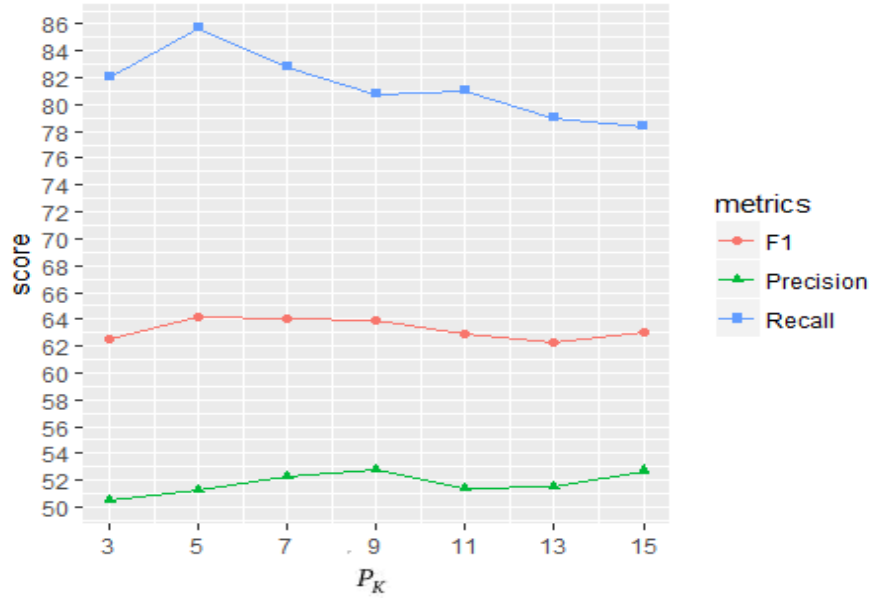


Figure 7.3: Precision and recall trends of K-means algorithm with different numbers of clusters.

$P_K$	Precision (%)	Recall (%)	F1 (%)
3	50.46	82.07	62.49
5	51.29	85.71	64.17
7	52.28	82.79	64.09
9	52.81	80.79	63.87
11	51.41	81.05	62.92
13	51.47	79.00	62.33
15	52.72	78.38	63.04

---

Table 7.5: Summary of results of K-means algorithm as a function of number of clusters.

---

#### Results for different dimensions of Word Embedding

---

We also ran experiments with three different dimensions of word embeddings. Table 7.6 shows that the performance can be improved with higher dimensions of word embeddings.

Dimension	Precision (%)	Recall (%)	F1 (%)
25	49.79	83.91	62.50
50	50.18	84.07	62.85
200	53.53	81.66	64.67

Table 7.6: Summary of results of K-means algorithm as a function of word embedding dimensions.

---

#### Optimal Parameter Settings for Candidate Words List

---

After observing the effect of various factors, we assume that the following settings can perform well for other datasets too.

$P_K=5$  ( $K$  is derived from this variable)

$C_{th}=3$

*Seeds List* = full list of seed words (MPQA Subjectivity Lexicon for English, SentiWS for German)

*Number of iterations* = 20 (initially fixed)

We applied these settings to Arab Spring and IGGSA dataset and compiled a list of candidate words for each one. The discovered words will be assigned polarities in section 7.3.

---

#### 7.2.2 Error Analysis

---

The performance of K-means algorithm can be impacted by certain errors made by the system. The threshold parameter  $C_{th}$ , which represents the minimum number of seed words that a cluster must have to be considered for a list of candidate words, causes the following two errors. For the first error, let's assume that a cluster has sentiment bearing words, but none of those words are present in the list of seed words. Though, the words in the cluster are relevant, the system will ignore the cluster and move on to checking the condition of the next cluster. That way, we can miss some relevant set of sentiment words. For the second error, let's assume that a cluster contains sentiment bearing words, but the inflected form of words in a cluster are different than the words present in the list of seed words. For example, a cluster contains the word "Angreifern" and our seed list has another inflected form of this word "Angreifers". In this case, the system will ignore the cluster, although the cluster contains the seed word but in another inflected form. This error can probably be solved by lemmatization. The system can get lemma of cluster words before matching them with seed words. Another error that can be made is that the system can not optimally minimize the sum of squared errors (SSE), which depends on initial cluster centroids. The system randomly finds the initial cluster centroids, which may be optimal or may not be, and can impact the performance of the algorithm.

---

Word embeddings can also have noise. Let's assume two words are accidentally seen together many times in a domain corpus. Word embeddings will learn this pattern and draw them close to each other in a vector space. However, one word is relevant and another word is not relevant for sentiment analysis, but both the words will end up on our candidate list because they are neighbors in a vector space.

---

### 7.2.3 Supervised Methods (CRFs and DRNN)

---

We also performed experiments with supervised learning, treating the task of detecting sentiment bearing words as a sequence labeling task. The reason for treating it as a sequence labeling task is we want to take context into account, i.e predecessor and successor of the target word. In this section, we present the results of experiments done with two different sequence labeling models, conditional random fields (CRFs) (Breck et al., 2007) and Deep Recurrent Neural Network (DRNN) (Irsoy and Cardie, 2014). To enable experimentation with these two machine learning algorithms, preprocessing of the training data and feature extraction is done first before training the algorithms and testing with different settings. We used sentiment labeled data (described in chapter 4), that consists of 9232 sentences annotated with “senti” and “non-senti” labels. Table 7.7 shows the example sentence from the dataset, having words labeled with “senti” and “non-senti” tags. Table 7.8 provides a summary of the dataset.

<b>Sentence</b>	The	issue	at	stake	was	apparently	simple
<b>Sentiment Label</b>	non-senti	non-senti	non-senti	non-senti	non-senti	senti	senti

Table 7.7: Example sentence from sentiment labeled dataset.

<b>Dataset</b>	<b>Number of Sentences</b>
Training	6462
Development	924
Test	1846

Table 7.8: Count of instances in each dataset.

---

### Feature Extraction and Representation

---

In machine learning, both the training and test data should be represented in some way in order to enable the algorithm to learn and build a model. We represent data based on features. Features are basically some attributes that are supposed to capture the pattern of the data. The entire data is represented in terms of these features before it is supplied to machine learning algorithms. In this section, we discuss the features used in our supervised algorithms. We used lexical features to capture specific words, part-of-speech features to learn the syntactic context, and word embedding features to capture various linguistic regularities and relationships of words.

- **Unigrams:** We include two predecessors and successors of the target word. For the current token which is defined to be at position  $i$ , we include  $\text{word}_{i-2}$ ,  $\text{word}_{i-1}$ ,...  $\text{word}_{i+2}$ .
- **Bigrams:** We include bigrams of neighboring words from unigram features.
- **Part-of-Speech:** We include a feature part-of-speech, defined to be the part of speech of the current token. We also include a part-of-speech tag of two predecessors/successors of the target word and bigrams and trigrams of neighboring part-of-speech tags from unigram features.
- **Word Embedding:** We include word embeddings trained on the dataset by the skip-gram model of word2vec.

## Conditional Random Fields (CRF)

We ran experiments for conditional random fields by using CRFSuite toolkit<sup>3</sup>. For our experiments, we chose a Stochastic Gradient Descent with L2 regularization term, sigma 0.32, and learning rate 0.32. L1 regularization biases are learned towards sparse solutions, and it is especially useful for high-dimensional problems with a large number of features. Tables 7.9 presents the results of experiments that we conducted with different combinations of features.

Features	Precision (%)	Recall (%)	F1 (%)
Part-of-Speech and Lexical Word features	71.93	64.56	68.04
Word Embedding	72.94	68.02	<b>70.40</b>
Word Embedding, Part-of-Speech, and Lexical Word Features	<b>76.82</b>	59.83	67.27

Table 7.9: Summary of results of CRF models trained with different combination of features.

The word embeddings alone used as features outperform others in terms of F-score, while the combination of word embedding, part-of-speech, and lexical word features achieves good precision out of all three. The system makes errors because of class imbalance dataset. The class “non-senti” outnumbers other class “senti” by a large proportion. The imbalanced dataset results in misleading accuracies and biased predictions.

We ran experiments with different settings of parameters of conditional random fields. In Figure 7.4, we observe the effect of sigma on the performance of CRF’s model. The figure shows that F-score declines when the sigma value goes up 0.32. The optimal value of sigma is 0.32.

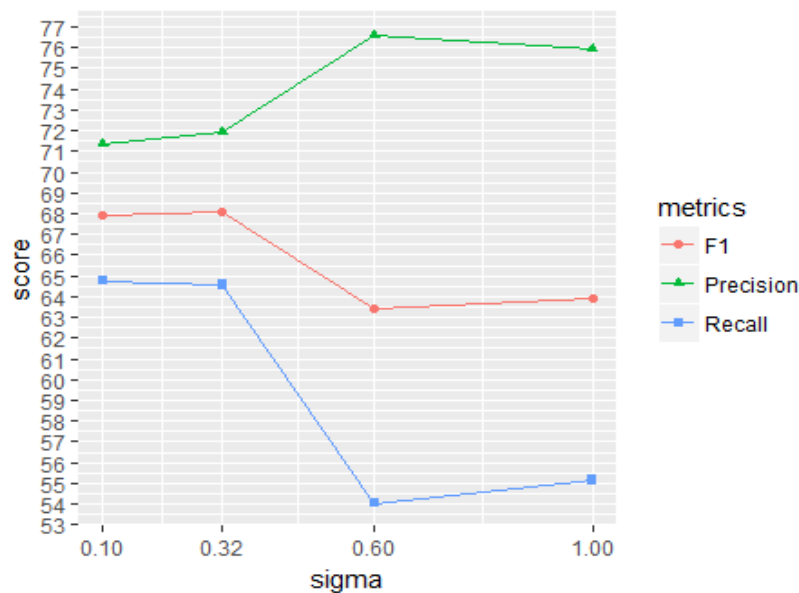


Figure 7.4: Precision and recall trends of CRF model with the varying values of sigma.

Embeddings scale is used for scaling factor when word vectors are added as features in the input of CRF’s model. Figure 7.5 shows that the optimal value of the embedding scale is 0.32. Table 7.10 shows that CRF model performed better with low dimensional word embeddings.

<sup>3</sup> URL:<http://www.chokkan.org/software/crfsuite/>

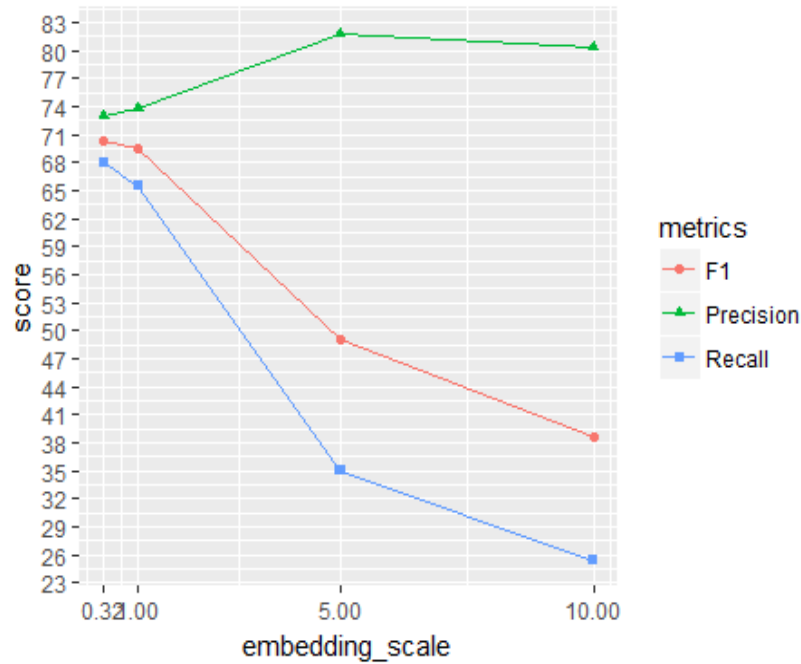


Figure 7.5: Precision and recall trends of CRF model with the varying values of embedding scale.

Word Embedding Dimension	Precision (%)	Recall (%)	F1 (%)
25	73.89	65.46	69.42
50	77.40	57.67	66.09
200	77.64	59.18	67.16

Table 7.10: Summary of results of CRF models trained with different dimensions of word embeddings.

## Deep Recurrent Neural Network (DRNN)

A recurrent neural network (Elman, 1990) is a neural network that has recurrent connections and memory. This feature makes them suitable for sequential prediction tasks where the interpretation of a single sentence is treated as analyzing a sequence of tokens. We used the implementation of Irsoy and Cardie (2014) for our experiments. Table 7.11 shows the results of DRNN method on the test set of sentiment labeled dataset.

Metrics	Score (%)
Precision	67.26
Recall	80.59
F1	73.33

Table 7.11: Results of DRNN method with 25 dimensional word embeddings.

## Conclusion

Table 7.12 displays the results of experiments with K-means clustering, conditional random fields (CRF), and deep recurrent neural network (DRNN), using the same testset and settings. We can see that super-



vised methods outperform the unsupervised method K-means clustering. The reason is that CRF's and DRNN effectively take into account the syntactic context of words in many forms (e.g. successors and predecessor of words, grammatical structure, etc.). The performance of K-means clustering is relatively low, because it extracts the words based on their proximity to seed words. It can happen that some words that do not possess any polarity are close to seed words by chance and picked up by the algorithm. As a result, the recall for K-means clustering is good, and precision and F-score are relatively poor.

However, the supervised methods need training data, which is generally not available for many domains and languages. Therefore, it seems best to use unsupervised methods with some trade-off with the F-1 score.

Methods	Precision (%)	Recall (%)	F1 (%)
K-means clustering	54.61	81.10	65.26
Conditional Random Fields (CRFs)	73.50	70.44	71.94
Deep Recurrent Neural Net (DRNN)	67.26	80.59	73.33

Table 7.12: Comparison of both supervised and unsupervised methods.

### 7.3 Experiments for Polarity Assignment

We evaluate the SENTPROP framework by recreating two known sentiment lexicons, MPQA Subjectivity lexicon and SentiWS. The MPQA lexicon is a list of 8,222 positive and negative labeled words (Wilson et al., 2005a). The SentiWS lexicon is a list of 1,650 positive and 1,818 negative labeled words (Remus et al., 2010a). For recreating MPQA lexicon, we used word embeddings trained on part of the Google News dataset (about 100 billion words). The model contains 300-dimensional vectors for 3 million words and phrases. We used 100 dimensional word2vec word embeddings trained on 116 Million German sentences for experiments with SentiWS. These resources are discussed in detail in Chapter 4.

We have chosen a minimal set of seed words as ideal examples of positive and negative sentiment words for German and English. These seed words are used commonly for expressing sentiments and are domain independent. The seed words for the German language are not in lemma form because we have chosen the inflected forms of words as they are used in IGGSA data set and ArabSpring corpus. If the words are in the lemma form, the system is not able to find some seed words in ArabSpring and IGGSA corpus, and then those seed words are not of any use.

<b>Positive Seed Words</b>	good, lovely, excellent, fortunate, pleasant, delightful, perfect, loved, love, happy
<b>Negative Seed Words</b>	bad, horrible, poor, unfortunate, unpleasant, disgusting, evil, hated, hate, unhappy

Table 7.13: English: Positive and Negative Seeds.

<b>Positive Seed Words</b>	schön, positiv, richtig, ausgezeichnet, erstaunlich, fantastisch, glücklich, perfekt, Liebe, Gut
<b>Negative Seed Words</b>	Schlecht, minderwertig, falsch, Verluste, Schäden, negativ, schrecklich, unglücklich, unangenehm, widerlich, gehasst

Table 7.14: German: Positive and Negative Seeds.

Section 7.3.1 presents the results of recreating known lexicons with SENTPROP. Sections 7.3.2, 7.3.3, and 7.3.4 explore variations on the SENTPROP system.

### 7.3.1 Recreating Known Sentiment Lexicons

We performed experiments with two traditional sentiment lexicons SentiWS and MPQA to evaluate the effectiveness of the SENTPROP approach. In our experiments, the parameters are set as follows: nearest neighbors  $nn = 4$ ,  $\beta = 100$ , and we use the seed words mentioned in Table 7.14 and Table 7.13. The accuracies of SentiWS and MPQA are listed in Table 7.15. The accuracy and precision are computed by comparing the predicted sentiment labels with the actual labels of words in SentiWS and MPQA lexicon. The result shows that the system achieved very good accuracy and performed well for English and German language when we recreated sentiment lexicons MPQA and SentiWS.

Sentiment Lexicon	Accuracy (%)	Precision (%)	F1 (%)
MPQA	88.96	81.73	75.89
SentiWS	81.13	82.59	73.93

Table 7.15: Performance metrics of recreating lexicons MPQA and SentiWS

In the following sections, we present the results of experiments performed with various settings of SENTPROP framework. All the following experiments recreated MPQA lexicon using Google news word embeddings with the different settings of model parameters.

### 7.3.2 Varying the Number of Nearest Neighbors (nn)

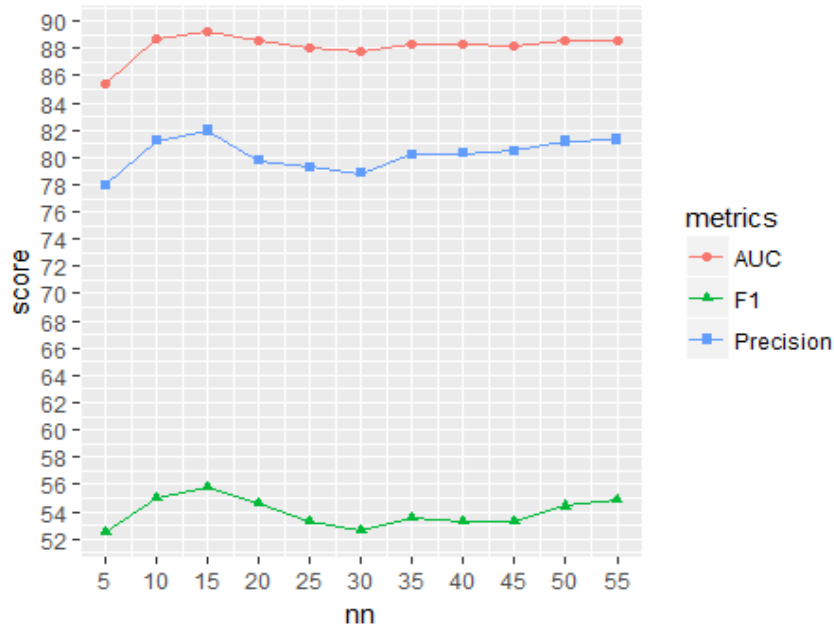


Figure 7.6: Accuracy and Precision trends of SENTPROP performance with the varying number of nearest neighbors (nn).

In SENTPROP, the label propagation algorithm first constructs a lexical graph by connecting each word to its  $nn$  nearest neighbors within the semantic space. In this section, we explore the impact of varying the amount of nearest neighbors ( $nn$ ). The parameter  $\beta$  is set to 0.99 for these experiments. Figure 7.6 shows accuracy as a function of nearest neighbors  $nn$ , as we vary the values from 5 to 55 words. From the graph, it seems best to have a small number of nearest neighbors, close to 15. The advantage of a small number of nearest neighbors is that words that occur close to each other are more likely to

be semantically related and would contribute more correctly to the final polarity of the target word. In the graph, we can see that when we increase the number of neighbors beyond 15, the accuracy starts to fall down because words that are far may provide incorrect polarity contributions to the final polarity score of the target word. The graph shows the accuracy is also poor with a very small number of nearest neighbors between 5 to 10. Thus, the optimal number of the nearest neighbors is between 10 to 20, likely near 15.

nn	AUC (%)	Precision (%)	F1 (%)
5	85.39	77.96	52.43
10	88.75	81.22	55.02
15	<b>89.27</b>	<b>81.97</b>	<b>55.84</b>
20	88.52	79.76	54.59
25	88.11	79.30	53.33
45	88.23	80.47	53.23

Table 7.16: The summary of performance of SENTPROP with the varying values of nn (Nearest Neighbor)

### 7.3.3 Varying the Parameter Beta( $\beta$ )

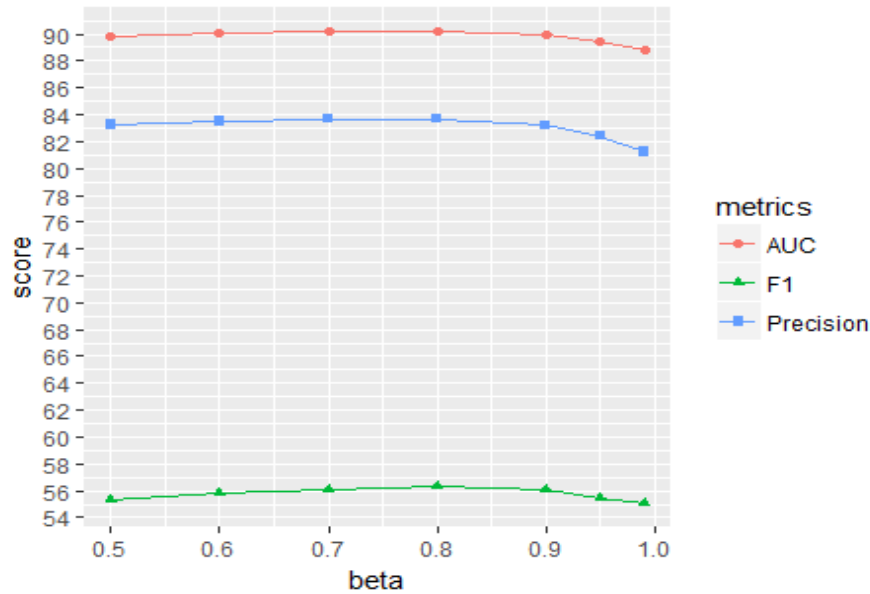


Figure 7.7: Effect of beta ( $\beta$ ) on the SENTPROP performance.

In this section, we investigate the effect of varying the parameter beta( $\beta$ ) in SENTPROP framework. Figure 7.7 shows the accuracy of the algorithm as a function of  $\beta$ . The parameter beta varies from 0.5 to 0.99, in an increment of 0.1. The parameter nn is set to 10 for these experiments. There are three curves in the graph for AUC, precision, and F1, and we see the same pattern of rising and fall for all the three curves. The beta parameter has a relatively little impact until it rises above 0.8, at which point the accuracy begins to fall off. The highest accuracy is achieved with beta 0.8. Second highest is beta 0.7, and the graph suggests that the optimal value of beta is somewhere between 0.6 and 0.9, likely near 0.8. The term  $\beta$  helps the algorithm to control the extent of favoring local consistency (similar labels for neighbors) versus global consistency (correct labels on seed words). The lower value of beta emphasizes the global consistency.

beta	AUC (%)	Precision (%)	F1 (%)
0.5	89.87	83.24	55.37
0.6	90.07	83.49	55.78
0.7	90.20	83.64	56.10
0.8	<b>90.23</b>	<b>83.65</b>	<b>56.38</b>
0.9	89.95	83.19	56.11
0.99	88.75	81.22	55.02

Table 7.17: Summary of results of SENTPROP with varying values of beta ( $\beta$ )

### 7.3.4 Varying the Number of Seed Words

The following experiment examines the behavior of SENTPROP with the increasing number of seed words. SENTPROP was designed to learn accurate domain specific sentiment lexicons using a small set of seed words (Hamilton et al., 2016). Figure 7.8 shows that the accuracy of the algorithm falls down constantly as we increase the number of seed words. The best accuracy was achieved when we used a set of eight seed words for each positive and negative category (section 7.3.1). It is clear that the SENTPROP framework performs much better with a small set of seed words.

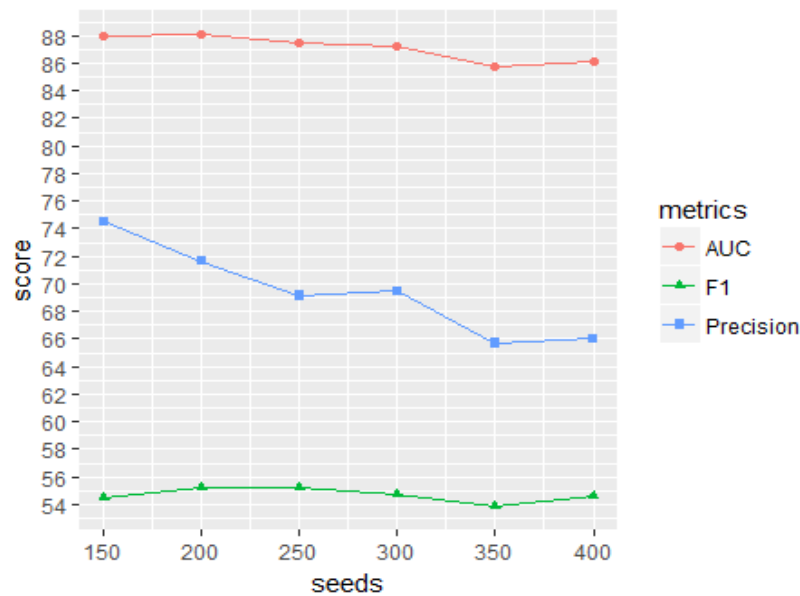


Figure 7.8: Effect of number of seed words on the performance of SENTPROP.

Number of Seed Words	AUC (%)	Precision (%)	F1 (%)
150	88.01	74.49	54.46
200	88.12	71.65	55.21
250	87.46	69.13	55.23
300	87.25	69.47	54.78
350	85.76	65.70	53.83

Table 7.18: The summary of the results of SENTPROP performance with varying the number of seed words.

---

### 7.3.5 Error Analysis

---

SENTPROP framework makes the following errors. The random walk algorithm propagates the labels from labeled nodes, which are positive and negative seed words, to unlabeled nodes. It happens that the system does not find all the seed words in a domain corpus. For example, our seed list has 10 positive and 10 negative words, but the system finds 6 positive and 8 negative seed words in a given corpus. This varying number of positive and negative seed words can influence the propagated polarity score. The reason is that the final polarity of a word is computed based on random walk visits from positive and negative seed words. Thus, if we use a generic seed list for all corpus, the system can make errors in propagating the scores. Another error the system can make is while assigning the positive and negative labels to words. The system assigns labels (positive or negative) based on the class-mass normalization method (Zhu et al., 2003). The class-mass normalization (CMN) method estimates a distribution of classes (how many positive, negative, and neutral words there are) from labeled data (labeled data can be any sentiment lexicon) and assigns labels to best match this distribution. The distribution of classes of labeled data, which is SentiWS in our experiments, does not fully represent the distribution in the generated lexicon. We conclude that both seed words and label assignment method can cause some bias in the results.

---

### 7.3.6 Discussion of Results

---

We validated the accuracy of the label propagation framework through various experiments. The system achieved accuracy of 88.96% on recreating MPQA lexicon (in English) and accuracy of 81.13% on recreating sentiWS lexicon (in German), which suggest that the approach can be extended to new languages. The number of seed words for SentProp seems to be an important parameter. A small set of seed words yields the best results. Additionally, for achieving best results, it seems that the value of the parameter *nn* (nearest neighbors) should lie in the range 10 to 15.

---

## 7.4 Extrinsic Evaluation: IGGSA Opinion Mining System

---

In this section, we evaluate the performance of our methods for generating domain specific sentiment lexicons with respect to the overall task of the opinion mining system which was proposed for German Sentiment Analysis Shared Task 2014. The German Sentiment Analysis Shared Task (GESTALT) is run under the guidance of Interest Group of German Sentiment Analysis (IGGSA)<sup>4</sup>, and it consists of two main tasks: Source, Subjective Expression and Target Extraction from Political Speeches (STEPS), and Subjective Phrase and Aspect Extraction from Product Reviews (StAR)(Ruppenhofer et al., 2014). The STEPS task involves extraction of subjective expressions, opinion sources, i.e. the entities that express an opinion, and opinion targets, i.e. the entities towards which an opinion is uttered, from German sentences. The sentences were taken from a corpus of political debates and speeches in the Swiss parliament (Schweizer Bundesversammlung). The opinion mining system that we use for our evaluation was built for STEPS task (Wiegand et al., 2016). It is a rule-based system which relies on a predicate lexicon specifying extraction rules for nouns, verbs, and adjectives. The rule-based system was made publicly available for allowing researchers to test different sentiment lexicons with different argument information about opinion sources and targets<sup>5</sup>.

The underlying assumption of the system is that the identification of opinion sources and targets is mostly determined by the opinion predicate (Wiegand et al., 2016). The system treats the task of ex-

---

<sup>4</sup> <http://iggsasharedtask2016.github.io/welcome.html>

<sup>5</sup> Opinion mining system for shared task 2014: <https://github.com/miwieg/german-opinion-role-extractor>

tracting opinion sources and targets as a lexical problem, and requires a lexicon of opinion predicates specifying the argument position of opinion targets and sources. The opinion predicates are the sentiment expressions that can be derived from sentiment lexicons. Sentiment lexicon is the most important resource for specifying opinion predicate rules and is the heart of the rule based system. This is, therefore, the main motivation for choosing this system for evaluation of our work. These opinion predicates can either be verbs, nouns, and adjectives. The extraction rules are designed for opinion predicates by taking into account their corresponding part of speech. The extraction rules are defined for verbs, nouns, and adjectives, as shown in Table 7.19.

Part of Speech	Source	Target
verb	subj	objd, obja, objc, obji, s, objp-*
noun	det, gmod	objp-*
adjective	author	attr-rev, subj

Table 7.19: Extraction rules for verb, noun, and adjective opinion predicates.

For instance, the rule for verbs assumes sources in subject and targets in object position. In the sentence (1), the sentiment is implied by the predicate **streiten**, the source is realized by its subject **Deutschland und die USA**, while the target is realized by its prepositional object **über das Freihandelsabkommen**. Thus, given a lexicon with argument information about sources and targets for each opinion predicate, the system checks each sentence for the presence of such opinion predicates and determines opinion sources and targets that are present in the sentence.

(1) Deutschland und die USA[source][subj] **streiten** über das Freihandelsabkommen [target][pobj].  
(Germany and the USA quarrel over the free trade agreement.)

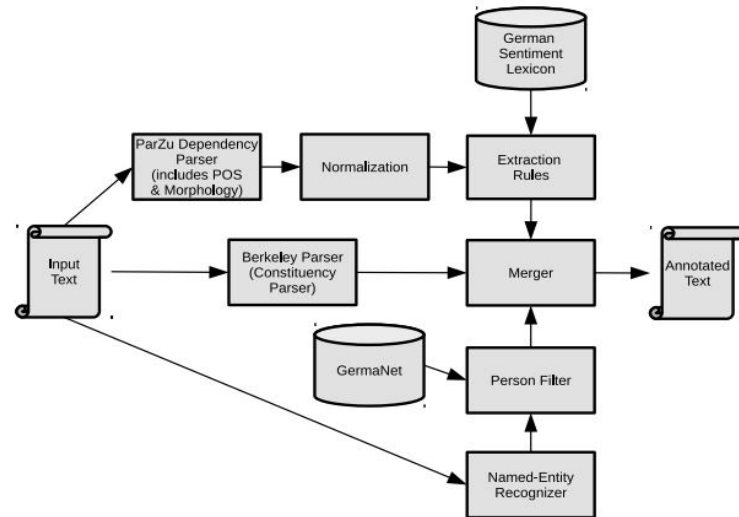


Figure 7.9: Workflow Pipeline of the Rule-based System (Wiegand et al., 2016).

Figure 7.9 shows the pipeline of the rule-based system. The pipeline employs dependency parser and constituency parser for obtaining linguistic information. Then, it carries out some normalization on the parsed output in order to have a more compact representation. The normalization includes, for instance, active-passive normalization for extraction rules that assume a sentence in active voice. After the linguistic processing, the system uses German sentiment lexicons for specifying the extraction rules.

The components i.e parser, extraction rules, and sentiment lexicon are the critical part of the system. The other component named-entity recognizer is used for detecting proper nouns and persons for some heuristics that says opinion sources must be persons. The output is the text annotated with Subjection Expression (SE), Source\_SE, and Target\_SE.

#### 7.4.1 Experiment

We performed the following experiments with rule-based system on the IGGSA dataset (discussed in Chapter 4 on page 22).

##### Run 1: Baseline

The baseline experiment employs a large sentiment lexicon which is the concatenation of PolArt and SentiWS lexicon. Table 7.20 shows the performance of the different configurations. SE evaluates the extraction of subjective expressions. Source SE evaluates the extraction of opinion sources given an exact match of subjective expression between the system output and the gold standard. Likewise, Target SE evaluates the extraction of opinion targets given an exact match of subjective expression between the system output and the gold standard.

Measure	SE	Source_SE	Target_SE
Prec (%)	36.07	57.86	75.77
Rec (%)	44.01	51.27	40.32
F-1 (%)	29.08	51.82	51.99

Table 7.20: Results of rule based system using a baseline lexicon: PolArt and SentiWS.

##### Run 2: Domain Specific Sentiment Lexicon

The second experiment employs the domain specific sentiment lexicon that we generated on the dataset by using our methods. The summary of the results is shown in Table 7.21. Table 7.22 reveals that, on the detection of sources (Source\_SE) and targets (Target\_SE) of subjective expressions, the domain specific sentiment lexicon outperforms the baseline (PolArt and SentiWS) lexicon, however, the latter performs better for detecting subjective expressions SE. There is a little difference between the precision of Source\_SE and Target\_SE for both the systems, but the domain specific sentiment lexicon produces better recall.

Measure	SE	Source_SE	Target_SE
Prec (%)	18.11	57.84	75.20
Rec (%)	47.11	59.45	40.22
F-1 (%)	26.07	<b>54.44</b>	<b>52.02</b>

Table 7.21: Results of rule based system using the induced domain specific sentiment lexicon.

Measure	Baseline	Domain Specific Sentiment Lexicon.
SE	<b>29.08</b>	26.07
Source_SE	51.82	<b>54.44</b>
Target_SE	51.99	<b>52.02</b>

Table 7.22: Comparison of F-score (%) of Baseline and Domain Specific Sentiment Lexicon



---

### 7.4.2 Error Analysis

---

When we used the domain specific lexicon, the system did not perform better than baseline on the detection of subjective expressions (SE). The reason is that the words in our generated sentiment lexicon are completely taken from the IGGSA dataset. The system can find subjective expressions in sentences for every opinion predicate in our lexicon regardless of whether they are right or wrong. As a result, the system finds a number of subjective expressions as large as a number of words in our generated lexicon. This is also the reason of low precision because not all subjective expressions found are correct. However, the baseline lexicon does not fully represent the vocabulary of the dataset. The system may not find some words from the baseline lexicon in sentences, resulting in less number of subjective expressions and better precision. For example, Table 7.23 shows the sentence from the gold labeled data. The subjective expression is “Effizienz” and its target is “der eingesetzten Mittel”. Table 7.25 shows the sentence from the baseline results, where the system found one wrong subjective expression. However, as shown in Table 7.24, two wrong subjective expressions were found in the system output with the domain specific lexicon. This shows that the system using a domain specific lexicon finds more subjective expressions because of exact matching of lexicon entries with the words in the sentences. This results in low precision of SE, Target\_SE, and Source\_SE, as the system finds targets and sources also for the wrongly found subjective expressions.

<b>Sentence</b>	Insgesamt	konnte	damit	die	Effizienz	der eingesetzten Mittel	gesteigert	werden
<b>Labels</b>					SE	Target_SE		

Table 7.23: The sentence from the gold labeled data.

<b>Sentence</b>	Insgesamt	konnte	damit die Effizienz	der	eingesetzten	Mittel	gesteigert	werden
<b>Labels</b>		Target_SE			SE	Target_SE	SE	

Table 7.24: The sentence from system results with the domain specific lexicon.

<b>Sentence</b>	Insgesamt	konnte	damit	die Effizienz der eingesetzten Mittel	gesteigert	werden
<b>Labels</b>				Target_SE	SE	

Table 7.25: The sentence from system results with the baseline lexicon.

In conclusion, we note that the rule-based system benefits from a domain specific sentiment lexicon. The precision of the system can be further improved when it is run with a big corpus. This is more likely due to the fact that better word embeddings can be generated on a big corpus.

---

## 7.5 Intrinsic Evaluation: Online User Survey

---

For intrinsic evaluation, we conducted online user surveys where we ask users to judge the polarity of the words of the induced sentiment lexicon. We had a set of five different surveys, where each survey has thirty different words. In the survey, we present a word along with three options, whether the word is positive, negative or neutral (or unknown to the user). Each survey was answered by three different users. The final polarity of the word was determined based on voting. Voting means if a word receives two user responses for positive polarity and one user response for negative polarity, then its polarity will be considered as positive. A sentiment lexicon was constructed for Arab Spring corpus. Table 7.26 shows few words from the generated lexicon.



Positive	Negative
zuverlässig	geldwäsche
bescheidene	unterdrückerische
populärer	falsch
engagement	überfällen
treu	beunruhigt
ausgezeichnet	ungerechtigkeit
lieber	nadelstichen
fördern	abhandenzukommen
schöner	bitter
fantastischer	protestszenen

Table 7.26: Few words from the generated lexicon for ArabSpring.

As a result of the surveys, we obtained 150 words with polarity labels for our evaluation data set. We have computed the accuracies with two different settings. In the first setting, we have only considered positive and negative words. The accuracy of the system is 79.84%. In Table 7.27, the confusion matrix provides the information such as how many positive words (determined by user responses) were predicted as positive and negative. The reason for using this setting is our system employs SentiWS as a seed lexicon, which does not have a neutral class. Thus, the system classifies all the words into two classes, positive and negative.

	Predicted Positive	Predicted Negative
(User Response) Positive	28	10
(User Response) Negative	6	34

Table 7.27: Confusion matrix with positive and negative class.

In the other setting, we also consider the neutral class for computing the accuracy. In table 7.29, the confusion matrix also provides the information about the neutral class, i.e. how many predicted positive or negative words were classified as neutral by users. The accuracy of the system with this setting is 41.33%. In the table, we can see that misclassification of the words as positive or negative is less and many words are judged by users as neutral. Therefore, we conclude that if the system is provided with a seed lexicon with all three classes, it can achieve good accuracy.

	Predicted Positive	Predicted Negative	Predicted Neutral
(User Response) Positive	28	10	37
(User Response) Negative	6	34	35

Table 7.28: Confusion matrix with positive, negative, and neutral class.

	Survey 1	Survey 2	Survey 3	Survey 4	Survey 5	Average
Krippendorff's alpha	0.56	0.549	0.605	0.312	0.471	0.5

Table 7.29: Krippendorff's alpha for all the surveys.

---

We calculated Krippendorff's alpha that provides measures of inter-coder agreement and inter-rater reliability. Krippendorff's alpha assesses the agreement achieved among annotators who categorize a given set of objects in terms of the values of a variable (Krippendorff, 2004). The average score of Krippendorff's Alpha for five surveys is 0.5. The Krippendorff's Alpha (inter-annotator agreement) is not low which means the users did not find it too difficult to agree on which words belong to the category and which didn't, and the definition of categories (positive, negative, or neutral) that needed to be annotated were clear. The disagreement mainly happened when one user voted for neutral class and others voted for either positive or negative class.

---

## 8 Conclusion

---

Sentiment lexicons are the most crucial resources for many opinion mining systems because they provide rich sentiment information. With more robust natural language processing (NLP) methods becoming available, the opinion mining systems flourished in various business domains and applications. Sentiment lexicons, upon which sentiment analysis often depend, are mostly developed for domain general context, and they can mislead sentiment analysis of specific domains and communities. As the sentiment of words is highly determined by context, opinion mining systems need an accurate and robust framework for automatically generating lexicons that are relevant to their domain.

The thesis presents methods for inducing domain specific sentiment lexicons. We developed an unsupervised algorithm, which extracts sentiment bearing words from a domain corpus and then assigns positive and negative polarity to the extracted words based on their usage in the domain context. The approach is implemented in a distributed environment, and can be extended to new languages or domains as it does not need any hand crafted lexical resources or labeled data. The methods have been tested for English and German. The system achieved accuracy of 88.96% on recreating MPQA lexicon (in English) and accuracy of 81.13% on recreating SentiWS lexicon (in German).

We validate the effectiveness and accuracy of our methods extrinsically and intrinsically with an opinion mining system and user surveys. We constructed a sentiment lexicon for IGSSA data set. Then, we applied this lexicon in the rule-based system built for IGSSA STEPS task to see if a domain specific sentiment lexicon plays a role in identifying opinion targets and sources. Experimental results on the benchmark dataset show that the F-score of the rule-based system on detecting sources of subjective expressions (Source\_SE) improved from 51.82% to 54.44%, indicating that the system can benefit from domain specific lexicons. Moreover, the results of the user surveys also show the good accuracy of 79.74%, when only positive and negative classes are considered. We have also done experiments with supervised methods (CRF's and Deep Recurrent Neural Network) for extracting the sentiment bearing words and compared the results with unsupervised method K-means clustering. The result shows that K-means clustering perform nearly with the same accuracy, but fits best for the approach as it does not need any labeled data. Experiments with different settings of word embeddings and seed words are done. We noticed that accuracy of K-means method highly depends on the used word embeddings, and label propagation approach is influenced by seed words. Therefore, the approach can be further improved by using word embeddings generated on bigger corpus and seed words provided by domain experts.

We believe sentiment lexicons generated by our methods can prove a useful resource for opinion mining systems and applications.

---

## 9 Future Work

---

The current study only focuses on individual words for sentiment lexicons. So, we can include the contextual polarity of phrases and idioms in the future work. For example, the customer review “These opera tickets cost us an arm and a leg”. The idiom “Costs an arm and a leg” is used when something is expensive. Although, individual words in this review do not express any positive or negative polarity, the sentence as a whole expresses a negative sentiment, because the idiom implies negative impact of the tickets cost. Therefore, it would be interesting to include idioms and phrases to a sentiment lexicon. We can also consider modifier words (e.g. very happy) and negation words (e.g. not bad) for the sentiment lexicons.

In the current study, we have done experiments with K-means clustering, which poses a challenge for finding the optimal value of  $K$ . In future, we can also conduct experiments with other clustering methods that do not require to define the number of clusters. In this work, we have noticed that word embeddings have an impact on the accuracy of the methods. It would also be interesting to experiment with other word embedding methods (Astudillo et al., 2015; Pennington et al., 2014) to see if different methods play a role in improving the system. Although, we have evaluated our work with IGGSA opinion mining system, we can also explore other established opinion mining methods (Hasan and Ng, 2013; Li and Hovy, 2014; Thomas et al., 2006) for further validation of our approach.

---

## Bibliography

---

- Astudillo, R. F., Amir, S., Ling, W., Silva, M., and Trancoso, I. (2015). Learning word representations from scarce and noisy data with embedding subspaces. In *ACL (1)*, pages 1074–1084.
- Asur, S. and Huberman, B. A. (2010). Predicting the future with social media. In *Web Intelligence and Intelligent Agent Technology (WI-IAT), 2010 IEEE/WIC/ACM International Conference on*, volume 1, pages 492–499. IEEE.
- Bar-Haim, R., Dinur, E., Feldman, R., Fresko, M., and Goldstein, G. (2011). Identifying and following expert investors in stock microblogs. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1310–1319. Association for Computational Linguistics.
- Breck, E., Choi, Y., and Cardie, C. (2007). Identifying expressions of opinion in context. In *IJCAI*, volume 7, pages 2683–2688.
- Coulouris, G. F., Dollimore, J., and Kindberg, T. (2005). *Distributed systems: concepts and design*. pearson education.
- Das, S. and Chen, M. (2001). Yahoo! for amazon: Extracting market sentiment from stock message boards. In *In Asia Pacific Finance Association Annual Conf. (APFA)*.
- Dave, K., Lawrence, S., and Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th international conference on World Wide Web*, pages 519–528. ACM.
- Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Communications of the ACM*, 51(1):107–113.
- Elman, J. L. (1990). Finding structure in time. *Cognitive science*, 14(2):179–211.
- Esuli, A. and Sebastiani, F. (2005). Determining the semantic orientation of terms through gloss classification. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 617–624. ACM.
- Fan, B., Tantisiriroj, W., Xiao, L., and Gibson, G. (2009). Diskreduce: Raid for data-intensive scalable computing. In *Proceedings of the 4th Annual Workshop on Petascale Data Storage*, pages 6–10. ACM.
- Hamilton, W. L., Clark, K., Leskovec, J., and Jurafsky, D. (2016). Inducing domain-specific sentiment lexicons from unlabeled corpora. *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.
- Hasan, K. S. and Ng, V. (2013). Stance classification of ideological debates: Data, models, features, and constraints. In *IJCNLP*, pages 1348–1356.
- Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 174–181. Association for Computational Linguistics.
- Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

- 
- Irsoy, O. and Cardie, C. (2014). Opinion mining with deep recurrent neural networks. In *EMNLP*, pages 720–728.
- Jia, W. and Zhou, W. (2004). *Distributed network systems: from concepts to implementations*, volume 15. Springer Science & Business Media.
- Joshi, M., Das, D., Gimpel, K., and Smith, N. A. (2010). Movie reviews and revenues: An experiment in text regression. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 293–296. Association for Computational Linguistics.
- Kamps, J., Marx, M., Mokken, R. J., and de Rijke, M. (2004). Using WordNet to Measure Semantic Orientations of Adjectives. In *In Proceedings of LREC-04, 4th international conference on language resources and evaluation, Lisbon, PT*, volume 4, pages 1115–1118.
- Klenner, M., Fahrni, A., and Petrakis, S. (2009). Polart: a robust tool for sentiment analysis. In *17th Nordic Conference on Computational Linguistics (NODALIDA 2009)*.
- Krippendorff, K. (2004). *Content analysis: An introduction to its methodology*. Sage.
- LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- Li, J. and Hovy, E. H. (2014). Sentiment analysis on the people’s daily. In *EMNLP*, pages 467–476.
- Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167.
- Loughran, T. and McDonald, B. (2011). When is a liability not a liability? textual analysis, dictionaries, and 10-ks. *The Journal of Finance*, 66(1):35–65.
- MacQueen, J. (1967). Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, Berkeley, Calif. University of California Press.
- Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *ICLR Workshop*.
- Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013b). Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.
- Mikolov, T., Yih, W.-t., and Zweig, G. (2013c). Linguistic regularities in continuous space word representations. In *HLT-NAACL*, volume 13, pages 746–751.
- Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., and Miller, K. J. (1990). Introduction to wordnet: An on-line lexical database. *International journal of lexicography*, 3(4):235–244.
- Mohammad, S. M., Kiritchenko, S., and Zhu, X. (2013). Nrc-canada: Building the state-of-the-art in sentiment analysis of tweets. *Atlanta, Georgia, USA*, page 321.
- O’Connor, B., Balasubramanyan, R., Routledge, B. R., and Smith, N. A. (2010). From tweets to polls: Linking text sentiment to public opinion time series. *ICWSM*, 11(122-129):1–2.
- Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up?: sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 79–86. Association for Computational Linguistics.

- 
- Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–1543.
- Reimers, N., Eckle-Kohler, J., Schnober, C., Kim, J., and Gurevych, I. (2014). Germeval-2014: Nested named entity recognition with neural networks. *Proceedings of the KONVENS GermEval Shared Task on Named Entity Recognition, Hildesheim, Germany*.
- Remus, R., Quasthoff, U., and Heyer, G. (2010a). Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*.
- Remus, R., Quasthoff, U., and Heyer, G. (2010b). Sentiws-a publicly available german-language resource for sentiment analysis. In *LREC*.
- Ruppenhofer, J., Klinger, R., Struß, J. M., Sonntag, J., and Wiegand, M. (2014). Iggsa shared tasks on german sentiment analysis (gestalt). In *Workshop Proceedings of the 12th Edition of the KONVENS Conference, Hildesheim, Germany, October. Universität Hildesheim*.
- Sadikov, E., Parameswaran, A., and Venetis, P. (2009). Blogs as predictors of movie success. Technical report, Stanford University.
- Sahlgren, M. (2005). An introduction to random indexing. In *Methods and applications of semantic indexing workshop at the 7th international conference on terminology and knowledge engineering, TKE*, volume 5.
- Schapire, R. E. and Singer, Y. (2000). Boostexter: A boosting-based system for text categorization. *Machine learning*, 39(2-3):135–168.
- Spark, A. (2015). Spark: lightning-fast cluster computing.
- Stone, P. J. and Hunt, E. B. (1963). A computer approach to content analysis: Studies using the general inquirer system. In *Proceedings of the May 21-23, 1963, Spring Joint Computer Conference, AFIPS '63 (Spring)*, pages 241–256, New York, NY, USA. ACM.
- Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307.
- Thomas, M., Pang, B., and Lee, L. (2006). Get out the vote: Determining support or opposition from congressional floor-debate transcripts. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 327–335. Association for Computational Linguistics.
- Tong, R. (2001). An operational system for detecting and tracking opinions in on-line discussions. In *Working Notes of the SIGIR Workshop on Operational Text Classification*, pages 1–6, New Orleans, Louisiana.
- Tumasjan, A., Sprenger, T. O., Sandner, P. G., and Welp, I. M. (2010). Predicting elections with twitter: What 140 characters reveal about political sentiment. *ICWSM*, 10:178–185.
- Turney, P. D. (2002). 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*, pages 417–424. Association for Computational Linguistics.
- Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.



- 
- Velikovich, L., Blair-Goldensohn, S., Hannan, K., and McDonald, R. (2010). The viability of web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 777–785. Association for Computational Linguistics.
- White, T. (2012). *Hadoop: The definitive guide*. " O'Reilly Media, Inc."
- Wiebe, J., Wilson, T., and Cardie, C. (2005). Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210.
- Wiebe, J. M., Bruce, R. F., and O'Hara, T. P. (1999). Development and use of a gold-standard data set for subjectivity classifications. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 246–253. Association for Computational Linguistics.
- Wiegand, M., Aliman, N.-M., Anikina, T., Carroll, P., Chikobava, M., Hahn, E., Haid, M., König, K., Lapp, L., Leeuwenberg, A., et al. (2016). Saarland university's participation in the second shared task on source, subjective expression and target extraction from political speeches (steps-2016). *Bochumer Linguistische Arbeitsberichte*, page 14.
- Wilson, T., Wiebe, J., and Hoffmann, P. (2005a). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT '05*, pages 347–354, Stroudsburg, PA, USA. Association for Computational Linguistics.
- Wilson, T., Wiebe, J., and Hoffmann, P. (2005b). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics.
- Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., Franklin, M. J., Shenker, S., and Stoica, I. (2012). Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing. In *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pages 2–2. USENIX Association.
- Zaharia, M., Chowdhury, M., Franklin, M. J., Shenker, S., and Stoica, I. (2010). Spark: cluster computing with working sets. *HotCloud*, 10:10–10.
- Zhou, D., Bousquet, O., Lal, T. N., Weston, J., and Schölkopf, B. (2004). Learning with local and global consistency. *Advances in neural information processing systems*, 16(16):321–328.
- Zhu, X. and Ghahramani, Z. (2002). Learning from labeled and unlabeled data with label propagation. Technical report, Citeseer.
- Zhu, X., Ghahramani, Z., Lafferty, J., et al. (2003). Semi-supervised learning using gaussian fields and harmonic functions. In *ICML*, volume 3, pages 912–919.