Supervised and Semi-Supervised Statistical Models for Word-Based Sentiment Analysis

Von der Fakultät Informatik, Elektrotechnik und Informationstechnik der Universität Stuttgart zur Erlangung der Würde eines Doktors der Philosophie (Dr. phil.) genehmigte Abhandlung

Vorgelegt von

Christian Scheible

aus Mühlacker

Hauptberichter:	Prof. Dr. Hinrich Schütze
Mitberichter:	Prof. Dr. Bernhard Mitschang

Tag der mündlichen Prüfung: 8. Juli 2014

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2014

Contents

Abstract

Ever since its inception, sentiment analysis has relied heavily on methods that use words as their basic unit. Even today, such methods deliver top performance. This way of representing data for sentiment analysis is known as the *clue model*. It offers practical advantages over more sophisticated approaches: It is easy to implement and statistical models can be trained efficiently even on large datasets. However, the clue model also has notable shortcomings. First, clues are highly redundant across examples, and thus training based on annotated data is potentially inefficient. Second, clues are treated context-insensitively, i.e., the sentiment expressed by a clue is assumed to be the same regardless of context. In this thesis, we address these shortcomings.

We propose two approaches to reduce redundancy: First, we use active learning, a method for automatic data selection guided by the statistical model to be trained. We show that active learning can speed up the training process for document classification significantly, reducing clue redundancy. Second, we present a graph-based approach that uses annotated clue types rather than annotated documents which contain clue instances. We show that using a random-walk model, we can train a highly accurate document classifier.

We next investigate the context-dependency of clues. We first introduce *sentiment relevance*, a novel concept that aims at identifying content that contributes to the overall sentiment of the review. We show that even when we have no annotated sentiment relevance data available, a high-accuracy sentiment relevance classifier can be trained using transfer learning and distant supervision. Second, we perform linguistically motivated analysis and simplification of a compositional sentiment analysis. We find that the model captures linguistic structures poorly. Further, it can be simplified without any loss of accuracy.

Deutsche Zusammenfassung

Eine der frühesten Methoden zur automatischen Sentimentanalyse nutzt Merkmalsrepräsentationen, die auf Wortvorkommen beruhen. Dieser Ansatz zur Datenrepräsentation ist der unter dem Namen *Clue-Modell* bekannt, da die Terme in einer größeren Spracheinheit *Schlüsselwörter* (Clues) für deren Sentiment sind. Das Clue-Modell ist noch immer einer der beliebtesten und erfolgreichsten Ansätze, da es einige praktische Vorteile gegenüber anderen Verfahren bietet: Es ist einfach zu implementieren und statistische Modelle sind mit einer solchen Repräsentation auch auf großen Datensätzen effizient trainierbar. Allerdings hat das Modell auch Nachteile. Erstens treten Schlüsselwörter redundant auf und kommen in vielen Trainingsbeispielen vor, so dass überwachtes Lernen ineffizient sein kann. Zweitens werden Schlüsselwörter kontextunabhängig behandelt, d.h., das durch einen Begriff ausgedrückte Sentiment ist unabhängig vom Kontext immer gleich. In dieser Dissertation stellen wir Lösungsansätze für diese beiden Nachteile vor.

Um Redundanz zu vermeiden, verwenden wir zunächst Active Learning, eine Methode des maschinellen Lernens, bei der das statistische Modell die Auswahl der Trainingsbeispiele vornimmt. Unsere Ergebnisse zeigen, dass wir durch Active Learning gleiche Klassifikationsgenauigkeit bei reduzierten Kosten erreichen, indem wir Redundanz zwischen Dokumenten vermeiden. Ein weiterer Ansatz zur Vermeidung von Redundanz beruht darauf, die Schlüsselwörter direkt zu annotieren. Annotierte Schlüsselwörter werden dann in einem graphbasierten Modell zur Dokumentenklassifikation verwendet. Wir zeigen, dass ein Random-Walk-Modell Dokumente mit hoher Genauigkeit klassifizieren kann.

Um die Kontextabhängigkeit von Inhalten zu bestimmen, führen wir die Idee der *Sentiment-Relevanz* ein. Als sentiment-relevant bezeichnen wir Inhalt, der zum Gesamtsentiment eines Dokuments beiträgt. Wir zeigen, dass wir selbst ohne annotierte Sentiment-Relevanz-Daten mit hoher Genauigkeit sentiment-relevanten Inhalt erkennen können. Dazu nutzen wir zwei Techniken des maschinellen Lernens: Transfer Learning und Distant Supervision. Zum Schluss untersuchen wir ein kompositionelles Modell zur Sentimentanalyse auf seine linguistischen Eigenschaften. Wir zeigen, dass das Modell nur schlecht linguistische Struktur erkennt. Zudem kann das Modell ohne Genauigkeitsverlust stark vereinfacht werden.

List of Tables

List of Figures

List of Abbreviations

AP	average polarity
BL	baseline
CEE	cross-entropy error
DNN	deep neural network
DS	distant supervision
HIT	human intelligence task
HITS	Hyperlink-Induced Topic Search
IMDb	Internet Movie Database
IMS	Institut für Maschinelle Sprachverarbeitung
L-BFGS	limited-memory Broyden-Fletcher-Goldfarb-Shanno
LA	label accuracy
MaxEnt	maximum entropy
MDS	Multi-Domain Sentiment Dataset
MinCut	minimum cut
MSE	mean squared error
MTurk	Amazon Mechanical Turk
NLM	neural language model
NLP	natural language processing
NN	neural network
P&L	Pang & Lee
PPR	Polarity PageRank
PR	PageRank
RAE	recursive autoencoder
SNR	sentiment nonrelevance
SR	sentiment relevance
TL	transfer learning
WA	worker accuracy

Acknowledgments

First of all, I would like to thank my advisor Prof. Dr. Hinrich Schütze. Without him, this thesis would not have been written. He gave me the freedom to explore and at the same time helped me find the right path. I would also like to thank my second reviewer, Prof. Dr. Bernhard Mitschang, and the chair, Prof. Dr. Sebastian Padó, for being part of the doctoral committee and for their helpful comments in the final stages of the thesis.

Thanks to all collaborators with whom I had the opportunity to work during the course of my PhD – Florian Laws, Lukas Michelbacher, Jimmy Dubuisson, and Jean-Pierre Eckmann – as well as to all the student research assistants of the D7 project: Onur Bazarkaya, Max Kisselew, Ina Rösiger, Beate Rothmund, Abdallah Salama, Svetlana Smekalova, Katja Wienicke, and Sihem Zidi.

I am thankful for all the helpful discussions with and comments from numerous colleagues. These are, first of all, the members of the sentiment analysis group – Khalid Al Khatib, Andrea Glaser, Charles Jochim, Wiltrud Kessler – and the deep learning group – Qi Han and Tobias Schnabel. Further, these include Anders Björkelund, Andre Blessing, Daniel Duran, Florian Laws, Thomas Müller, Kyle Richardson, Stephen Roller, Wolfgang Seeker, and Jason Utt, as well as probably many others who I regretfully fail to recognize here. For proofreading, I am indebted to Andre Blessing, Charles Jochim, Wiltrud Kessler, Kyle Richardson, and Jason Utt. Special thanks go to all my office mates – Wiltrud Kessler, Andrea Glaser, Jason Utt, and Abhijeet Gupta – for a great time, as well as to Charles Jochim for being the best next-door neighbor one could wish for.

I am grateful for having had the opportunity to work at Sony while writing the thesis, made possible by Thomas Kemp. The work there was interesting and challenging, and it made for a more than welcome distraction from writing.

Lastly, I would like to thank my family for always being there and for not asking any questions. And Nina for bearing with everything.

The work in this thesis was supported by the DFG as part of the Sonderforschungsbereich 732 “Incremental Specification in Context”.

1 Introduction

The open exchange of opinions is becoming increasingly important The world wide web, for example, is an endless resource of opinions. People frequently express their opinions using online platforms designed for reviewing, or publish their thoughts on the social web. With the availability of information in large quantities comes the need for browsing it efficiently. For example, a popular product on the online retail site Amazon.com can have several thousand reviews; the Playstation 4 video game system had already been reviewed over 3,000 times a week after its release.[1] For a reader, it is impossible to read all reviews to find out about commonly pointed out up- or downsides of a product. Examples like this motivate *sentiment analysis*, the automatic analysis of opinions expressed through natural language.

The concept of sentiment analysis covers a wide range of interesting problems that are related to the way humans express emotions and opinions about various entities. Sentiment has been in the focus of natural language processing (NLP) research for a long time. Early computational efforts were made on the automatic recognition of the sentiment on the word-level, on rule-based analysis of sentiment on the web, as well as on the detection of subjective expressions. The classical sentiment experiment that arguably had the most impact in both the NLP and machine learning communities was the automatic prediction of the overall sentiment expressed in reviews with simple statistical machine learning models. These results paved the way for many approaches to automatic sentiment classification on user-generated data.

In this thesis, we will follow the latter line of work. The general task that we address is *sentiment classification*, the prediction of the *polarity* (i.e., the degree of positivity or negativity) of a complete linguistic unit. We tackle this problem through statistical natural language processing, where a statistical model is trained to automatically make predictions about unseen examples. The linguistic units that we analyze are either sentences or documents containing expressions of sentiment.

In statistical natural language processing, the way data is formally represented is

[1] `http://www.amazon.com/dp/B00BGA9WK2`

crucial. In sentiment analysis, this question is particularly important as sentiment is highly compositional and can be expressed through arbitrarily complex linguistic structures. The overall sentiment expressed by a phrase depends on the basic expressions it contains and the way they interact. There are cases where individual words or phrases act as clues (e.g., *awesome*) that are sufficient to detect sentiment, but in others, the overall sentiment will be determined by composition of meaning (e.g., *not awesome at all*). This view has been formally captured by two models: the *clue model*, which is the underlying method of the approaches mentioned above, where the words in a larger linguistic unit are viewed as clues that give away the overall polarity of the unit; and the *polarity shifter model* where the polarities of clues are (recursively) changed by modifying expressions. We will describe these models in more detail in Chapter 2.

The clue model is a popular choice for many sentiment analysis tasks. It is straightforward to implement and keeps the required learning effort low. If enough data is supplied to the model, it also works very well, leading, for example, to high classification accuracies for the detection the overall sentiment in a document. In contrast, the polarity shifter model is difficult to learn as the set of possible modifiers (and modifying construction types) is open. For this reason, work on sentiment classification often resorts to the clue model exclusively. Thus, researching this model is of high interest in both theory and practice. While, as mentioned above, the clue model achieves reasonable results, there are also several drawbacks to it. We will point out two major issues of the model which we will address in this thesis.

Redundancy Clues will be highly redundant across examples. This observation is related to the well-known power-law distribution of words, as described for example by Zipf (1935). As a consequence, there will be a small number of clues that appear in many examples, as well as a large number of rare clues. For example, words like *good* or *bad* will be frequent, and thus will also be sufficient indicators for many examples. Words like *delightful* or *marvelous* are rarer. However, to improve the results any further – particularly by covering short examples – we require information about rare clues. We pursue two possible approaches that address this problem. Both aim at efficiently extending the set of clues about which we have

knowledge. In the following, we imagine a supervised statistical modeling scenario where we have training data annotated by humans. As human labor is expensive, annotations cause costs.

As a first attempt to increase coverage, we could obtain more training examples (e.g., documents or sentences). In this case, the way we select new examples is crucial. If we employ manual annotation, any redundancy will lead to unnecessary additional costs. Thus, we want new examples to be as informative as possible. As we will later show, this process can be addressed through active learning, a machine learning technique where the learner decides which examples are selected for annotation.

Annotating whole examples to improve the clue model is in a sense backwards. Unlike other natural language representations, clues can easily be annotated directly by humans. For this reason, we are interested in a statistical model that enables us to use knowledge about clue polarity. However, including clue knowledge into a document model is not straightforward. We propose an approach where clues and documents are jointly represented in a graph.

Context Insensitivity By design, the clue model works by assigning a polarity to each word independently of its context. As mentioned before, this assumption is convenient as it drastically simplifies statistical clue models. However, in reality, the polarity of a word can change, depending on the context in which it occurs. For example, the adjective *light* can have different polarities depending on the object it describes. In theory, the distinction between prior polarity, i.e., the polarity that a word has without any context, and contextual polarity has been proposed. Recognizing contextual polarity is difficult as context effects can arise from different domains, topics, and objects being described. This implies that a large amount of supervision is needed to make contextual polarity universal. Thus, to keep matters simple, in many applications, each clue is taken to always indicate the same polarity, regardless of its context. Naturally, this approach may lead to the misinterpretation of clues, which could potentially hurt the model's prediction accuracy.

One way to address the problem of contextuality is to view its recognition as a pre-processing step. In this thesis, we introduce the concept of *sentiment relevance*

to address topic effects. The goal is to distinguish contexts that are important for sentiment detection from unimportant ones.

As mentioned previously, it has also been proposed to address the problem of contextuality through polarity shifters. As shifters are difficult to learn automatically, fully compositional approaches have been proposed. These models promise a holistic approach to sentiment analysis as we do not have to worry about modeling the detection of modifiers. However, their properties are poorly understood. We will provide detailed analysis of a compositional neural network model with respect to its linguistic interpretability and to the contribution of the structures it produces.

Finally, we are interested in efficient solutions. As one of the main motivations of the clue model is its simplicity, we would like this property reflected in our approaches. Thus we aim for methods that improve the clue model while keeping both computation and utility costs low.

1.1 Contributions in this Thesis

In this thesis, we make the following contributions concerning the two challenges discussed above.

Redundancy

- We show that we achieve high-accuracy polarity prediction through active learning even with noisy annotations. We perform document classification experiments on Amazon Mechanical Turk in which active learning outperforms random example selection significantly. Using active learning, the cost for achieving a certain level of accuracy can be lowered significantly. Qualitative analysis of the feature weights of the classifiers over time shows that useful features are identified faster using active learning than with random selection (Laws et al., 2011).

- We introduce a graph framework for joint classification of words and documents. We show that using a random walk algorithm, polarity information for words can be exploited to classify documents more accurately than with

a baseline averaging method. To this end, we adopt Personalized PageRank, calling the resulting algorithm Polarity PageRank. Finally, we show that a maximum entropy classifier trained in a subsequent bootstrapping step improves the results further (Scheible and Schütze, 2012).

Context Insensitivity

- We introduce the concept of sentiment relevance. It aims at identifying content that does not contribute to the overall sentiment in a document. We first contrast sentiment relevance to subjectivity, a related concept, through annotation and classification experiments. We next introduce two methods for automatic sentiment relevance prediction, distant supervision and transfer learning. Distant supervision makes use of a domain-specific database to create an initial annotation for unlabeled examples. Transfer learning learns from a related concept, subjectivity, for which labeled data is available and uses adaptation techniques to apply this knowledge to sentiment relevance. These approaches are supported by a graph-based unsupervised sequence model. We show that both significantly outperform the respective baseline methods (Scheible and Schütze, 2013b).

- We present an analysis of the Semi-Supervised Recursive Autoencoder (RAE), a compositional neural network model. The RAE takes a sentence as its input and constructs a tree where each node has a feature vector. The vectors of the tree are used to predict the sentiment of the sentence, which can influence the vectors further. As these structures are generated automatically and semi-supervisedly, their interpretation is unclear. We perform two experiments to analyze them. First, we have humans check the structures for syntactic and semantic coherence. We find that neither of these properties are reflected well in the RAE trees. Second, we introduce methods for automatic tree simplification. We show that only part of the RAE trees actually contribute to high-accuracy sentiment classification (Scheible and Schütze, 2013a).

Efficiency

- In all our approaches, we use efficient methods for computation. Most of our experiments build on either maximum entropy models with fast optimization or efficient graph algorithms. In case of the costly compositional neural network model, we show that it can be simplified structurally, which speeds up computation.

1.2 Structure of this Thesis

The remainder of this thesis is structured as follows.

We first introduce the background that we make use of in this thesis. In **Chapter 2**, we give an overview of statistical natural language processing. We focus on the general ideas as well as the different models that we apply throughout the thesis. **Chapter 3** introduces concepts relevant to sentiment analysis that are important to the understanding of this thesis.

We then turn to the four empirical studies introduced above. **Chapter 4** is concerned with active learning. We first describe our annotation setup on Amazon Mechanical Turk. We then run experiments and discuss the results, including an analysis of the features learned by the classifier.

In **Chapter 5**, we present our work on graph-based sentiment classification with Polarity PageRank. First, we describe the Polarity PageRank method. Second, we conduct experiments using graphs constructed with data from Wikipedia and Amazon.com. We perform error analysis taking into account the structural properties of the graph.

Chapter 6 is centered around the concept of sentiment relevance. We introduce a sentiment-relevance annotated dataset and describe the annotation process. We next show experimentally that the concept differs from related notions. We then present experiments on two approaches to automatic sentiment relevance classification.

Chapter 7 contains experiments on compositional sentiment analysis with the RAE model. We first describe the model, pointing out important implementation details. We introduce several ways of simplifying the tree structures the model

induces. Finally, we present the results of two experiments: human analysis of RAE trees and automatic tree simplification.

Chapter 8 concludes the thesis and provides an outlook on future work.

In **Appendix A**, we describe the tools and datasets used in the thesis in more detail.

2 Statistical Natural Language Processing

Over the years, statistical modeling has become the dominant approach to natural language processing. It has lead to broad-coverage applications capable of handling vast amounts of data and is responsible for substantial improvements in various NLP tasks such as part-of-speech tagging, syntactic parsing, and word sense disambiguation (Norvig, 2011). As an example for a successful practical application of statistical NLP, IBM recently demonstrated how large-scale combination of statistical systems can outperform humans in question answering.[2] We will now give a brief overview of the general ideas behind statistical modeling and how it relates to NLP.

Hastie et al. (2001) define a *statistical model* as a mathematical model, i.e. a function $f : x \mapsto y$ that predicts an output y for a given x. If y is categorial, we call f a *classification* function. The function f can be found for example by learning from some given input data for which the true output is known. Norvig (2011) provides a more general view. He proposes to distinguish the following properties of statistical models:

- A **mathematical** model, defined through a mathematical function $f : x \mapsto y$.
- A **probabilistic** model, where a probability involving properties of x and y is estimated.
- A **trained** model, which is selected or improved by using a collection of examples. This selection may be performed through the adjustment of parameters within the model.

A model qualifies as statistical if any of these properties apply. Although current research is dominated by models that are at mathematical and trained (and often

[2] `http://www.ibm.com/watson`

probabilistic), we can see that the term "statistical model" covers a large variety of different types of models.

The application of such models to linguistic problems has a longstanding history. The origins of statistical models of language go back to the *mathematical theory of communication* of Shannon (1948). Many of the ideas expressed in Shannon's work, such as n-gram models of language, are still popular today. They inspired the development of powerful language models and lead to many efforts for formally representing natural language through *features* to make them accessible to statistical models. The process of finding good feature representations for a task is called *feature engineering*.

Mitchell (1997) provides a well-known formal definition for the learning aspect of statistical models, the *machine learning* problem:

> A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.

This abstract formulation of machine learning raises an important question. How can we use data to improve the quality of a statistical model? One answer to this question comes in the form of supervised, semi-supervised, and unsupervised learning, which we will look at in more detail in Section 2.1. In Section 2.2, we will take a look at how language can be represented formally as input to a statistical model. Section 2.3 describes several learning strategies that can be used to improve or simplify the model learning. We next introduce the statistical modeling approaches we use, focusing on discriminative models (Section 2.4) and graph-based approaches (Section 2.5). In Section 2.6 we describe the evaluation measures, hypothesis tests, and agreement measures that we apply in this thesis.

Notation During this thesis, we will use the following notation. Row vectors will be typeset as lowercase bold letters (e.g., $\mathbf{v} = (v_1, \ldots, v_n)$ with elements v_i), matrices as capitalized bold letters (e.g., $\mathbf{M}$ with elements m_{ij}). $\mathbf{1}$ is an all-ones vector, and $\mathbf{0}$ an all-zeros vector. $\mathbf{v}^\intercal$, a column vector, denotes the transpose of $\mathbf{v}$. $[\mathbf{u}, \mathbf{v}]$ denotes the horizontal concatenation of two vectors $\mathbf{u}$ and $\mathbf{v}$, $[\mathbf{u}; \mathbf{v}]$ their vertical concatenation (a matrix with two rows). Horizontal concatenation can also

be expressed with the operator $\oplus$. Sets will be written as calligraphic symbols, e.g., $s \in \mathcal{S}$. We will refer to the set of classes c assigned by a model as $\mathcal{C}$. Sometimes, we need to distinguish instances of variables for different types t, which we express through a superscript index, e.g., $\mathbf{w}^{(t)}$.

We write the set of n examples as $x_k \in \mathcal{X}$ and the corresponding outputs as $y_k \in \mathcal{Y}$. The set of observed examples is denoted by $\hat{\mathcal{X}}$, the set of target outputs by $\hat{\mathcal{Y}}$. The (labeled) data $\mathcal{D} = [\langle \hat{x}_i, \hat{y}_i \rangle | 1 \leq i \leq n]$ is ordered, consisting of pairs of examples and their targets. Examples and labels can be converted into a vectorized representation where examples are be represented by the vector $\mathbf{x}_k$ and the corresponding target outputs by $\mathbf{y}_k$, which is a $|\mathcal{C}|$-dimensional vector where each dimension corresponds to a $c \in \mathcal{C}$. A collection of n examples together can be viewed as a matrix $\mathbf{X} = [\mathbf{x_1}; \ldots; \mathbf{x_n}]$ where each row contains a vector representing an example, and the target outputs as $\mathbf{Y} = [\mathbf{y_1}; \ldots; \mathbf{y_n}]$.

The goal of a statistical model is to learn a function $f_{\mathbf{W}} : x \mapsto y$, where $\mathbf{W}$ is a matrix which determines the behavior of f. In cases where f outputs a probability, we can write it as $p_{\mathbf{w}}$ or simply p.

2.1 Supervised, Unsupervised, and Semi-Supervised Models

Machine learning is the problem of finding the function f. There are two major paradigms for learning f inductively: supervised and unsupervised learning.

In *supervised learning*, we assume that we have observed data, the *training data* or *labeled data* $\mathcal{D}$, consisting of some examples $\hat{\mathcal{X}}$ with (correct) corresponding outputs $\hat{\mathcal{Y}}$. We can then learn f as a trained model, applying the machine learning principle. In most cases, this means that we minimize some measurement of the error the model makes on this data. For mathematical models, this can be accomplished for example by numerically optimizing the model parameters.

In *unsupervised learning*, we are not given any information about $\hat{\mathcal{Y}}$. This means that we not only do not know the correct outputs for some given examples $\hat{\mathcal{X}}$ (commonly referred to as *unlabeled data*), but in some cases, we may have incomplete or no knowledge about the set of possible labels $\mathcal{C}$. Instead, f will map an example to

a set of anonymous classes, whose cardinality we may have to determine by some other means. The anonymous classes are usually referred to as *clusters*. Since we cannot measure the error of f in terms of prediction, we have to resort to other means of determining its quality. For example, we can maximize the likelihood of the data under the model.

Semi-supervised learning combines elements of supervised and unsupervised learning. As the name implies, we are given some information about $\hat{\mathcal{Y}}$, while some other part may be missing. Among other possible scenarios, this may mean that we have correct outputs for some but not all of the data. We can then try to leverage additional information from the unlabeled data in combination with supervised learning on the labeled data. While this is probably the most prominent setup in semi-supervised learning, it is not the only one. Others include missing information, such as incomplete sequence order, and noisy or mismatched input data.

2.2 Representing Language for Statistical Modeling

In many applications of machine learning it is necessary to pre-process data to make it accessible to the model. In natural language processing, the data is usually a piece of text, i.e., a string of symbols from a fixed alphabet (e.g., all characters defined in unicode). Human readers can interpret these sequences as words, which may be marked as such through spaces in the text in some languages. It has been assumed that readers then interpret a sequence of words through hierarchical composition, constructing the meaning of phrases and sentences incrementally (Chomsky, 1965).

Conversely, many popular statistical models work with vector inputs, i.e., data consisting of distinct dimensions, called *features*, which are each associated with a quantity. Features are often assumed to be conditionally independent, i.e., the co-occurrence of two features does not change their interpretation. This contradicts the aforementioned process of how humans understand language. For this reason, the way in which language is converted into a feature representation is crucial

to the success of any machine learning approach. This process is called *feature extraction*.

Formally, $\phi_i(x)$ is a *feature function* which quantifies a specific property (indexed by $i \in \mathcal{F}$, the set of possible features) of an example x:

$$\phi_i(x) = \text{the number of times example } x \text{ has feature } i$$

If we represent the featurized data as a vector $\mathbf{x}$, we can populate this vector with the feature function by mapping each f to a dimension of $\mathbf{x}$. For some models such as the maximum entropy model which we introduce in Section 2.4, the function has to be strictly binary-valued to encode the presence or absence of the feature.

A simple and popular approach to feature extraction for representing texts is to assume that word order does not matter. We interpret a text x as a set of its words $w \in x$ and ignore the order in which they occurred. This approach is known as the *bag-of-words model*, since we can view the process as taking all words from the text and throwing them in a bag, losing sequence information in the process. We obtain the (binary) bag-of-words model through the following feature function:

$$\phi_i(x) = \begin{cases} 1 & \text{if } x \text{ contains word } w_i \\ 0 & \text{else} \end{cases}$$

The bag-of-words representation assumes that it is sufficient to use individual words as indicators. The model can easily be extended to extract arbitrary symbolic parts from the text. For example, one could define a feature function that extracts all word sequences of length n, i.e., all *n-grams*:

$$\phi_{\langle w_1,\ldots,w_n\rangle}(x) = \begin{cases} 1 & \text{if } x \text{ contains the word sequence } w_1 \ldots w_n \\ 0 & \text{else} \end{cases}$$

Feature extraction is a crucial step in natural language processing, so it is not surprising that it has been the focus of many publications. Linguistically motivated approaches have been proposed, employing a rich pre-processing pipeline including part-of-speech tagging and syntactic parsing among other steps. Feature functions are then defined manually to heuristically extract information from the resulting

structural analysis.

Ultimately, the feature extraction paradigm might not be sufficient to cover all complex phenomena that occur in natural language. For this reason, many improvements have been proposed on the model side, with the goal of handling language structure within the model.

One popular approach uses kernel methods in order to enable existing models to handle structured data whose explicit enumeration would be intractable. For example, tree kernels (Moschitti, 2006) for support vector machines are capable of performing efficient pairwise comparisons of syntactic structures.

More recently, models combining vector space semantics and supervised machine learning have emerged. The basic idea behind these models is that the meaning of words and phrases are encoded in some vector space which serves as the domain of a supervised classifier. Embeddings of words into this space are learned automatically, taking the subsequent prediction task into account. Thus, this area of research is also known as *representation learning*. There, neural network models of language have enjoyed much success, such as neural language models which we introduce in Section 2.4.2.

2.3 Learning Strategies

In machine learning, the way the learner approaches the problem can play a decisive role. The data used for training a model is particularly important as it directly influences what the model learns. We will first introduce two strategies to iteratively improve a statistical model, active learning and self training. We then present two strategies for semi-supervised learning that circumvent the need for customly produced manually annotated training data.

Active Learning

Active learning (AL) is a machine learning technique for joint data annotation and training of a supervised classifier. The general active learning process is viewed as a cycle of interaction between the annotator and the classifier. First, the classifier is trained on an already labeled data-set. It can then then request specific additional

input from the annotator. With this new information, the system is re-trained and the active learning cycle starts again. Active learning potentially lowers the overall cost of annotation since the system can ask for specific information it requires to improve itself. We will introduce the basic active learning concepts that we apply based on the survey by Settles (2009).

Uncertainty selection with pool-based sampling (Lewis and Gale, 1994) is a simple implementation of the AL idea. Here, the cycle is designed as follows: First, we annotate a small set of examples as the initial training set. In addition, we collect a large *pool* of unannotated examples. We then train a supervised classifier on the training set. This classifier is applied to the examples in pool. For each example in the pool, the classifier calculates how *uncertain* it is about its prediction. We then take strongly uncertain examples, have them annotated, and add them to the training set. Finally, the classifier is re-trained. The intuition behind this process is that the most uncertain examples will be maximally informative for the classifier. In order to perform uncertainty selection, we need to define a way of measuring uncertainty. and a selection criterion based on this uncertainty measure.

We assume that our classifier can estimate a probability $p(c|x)$ for a class c given a example x. Ranking the classes by decreasing probability $p(c_i|x)$, where $(c_1, \ldots, c_n)$ is the ranked list of classes, we obtain the *margin* metric (Scheffer et al., 2001)

$$M(x) = p(c_1|x) - p(c_2|x)$$

which calculates the difference in probability between the two most probable classes. The simplest way of making use of this metric is to select the most uncertain example, i.e., the example with the smallest margin, $x = \arg\min_x M(x)$, annotate it and re-train. However, re-training might be an expensive process in some applications, so we would like to avoid it if possible. *Batch* annotation addresses this problem by selecting the n most uncertain examples for annotation in one AL iteration. Batch selection might however yield a subset of redundant examples (Brinker, 2003). It is thus important that new labels are incorporated quickly as otherwise, available information is not used and we might annotate redundant examples. *Staleness* (Haertel et al., 2010) quantifies this effect as the number of new annotations we have obtained since the classifier was retrained.

Self-Training and Bootstrapping

Self-training (also referred to as *bootstrapping*) is a machine learning technique for increasing the amount of training data for a classifier through automatic annotation of unlabeled data. In a typical setting, a small set of labeled data and a large set of unlabeled data are available, which is commonly referred to as the *pool*. A classifier is trained on the labeled data and applied to the unlabeled data. We can then take a subset of the automatically classified examples for which the classifier has the highest confidence and add them to the training set. If the confidence measure is reliable, these examples should likely be correct and after re-training, the quality of the classifier should increase.

In its basic setup, this technique is similar to the previously introduced active learning – the classifier guides the selection of examples from a pool. In contrast, however, self-training selects the most confidently classified examples rather than the most uncertain ones since we will use the predicted label directly instead of consulting a human annotator.

Self-training can be applied iteratively. After selecting examples from the pool, adding them to the training set and re-training the classifier, we can again classify the remaining examples of the pool to improve the labeling. Note that it is possible too keep examples in the pool even after we added them to the training set. This is sometimes referred to as *sampling with replacement*. The intuition is that if a selected example has been misclassified previously, we might be able to correct this mistake later on with a better classifier.

The self-training process may be initialized in several ways. A straightforward idea is to start with a small set of labeled examples to produce an initial classifier to automatically label the pool. This classifier is typically a statistical model and it is the one that will be improved through self-training.

Another approach is to provide an initial classifier, which we will refer to as the *base classifier*, instead of an initial labeled dataset. This version of self-training works without having to label any data in order to start the self-training process. Thus, an arbitrary classifier can be used to perform the initial labeling, such as a rule-based classifier. Note, however, that the base classifier needs to be capable of estimating confidence in order to select an initial set of training examples.

Distant Supervision

Distant supervision (DS, Mintz et al., 2009) is a machine learning technique that makes use of *weak indicators* in order to learn to classify examples. A weak indicator can be any source of information that is annotated with the same set of labels as the examples for which we want to make predictions.

The typical distant supervision setting uses a database of indicators for training. In addition, a pool of unlabeled examples is required. First, the database of indicators is used to annotate the examples in the pool. The way this annotation is accomplished is open. For example, the database could contain labeled features that match the examples. If multiple database entries match an example, a decision scheme is needed to select a label; occurrence counts or probability estimates are frequently used (e.g., Go et al., 2009; Surdeanu et al., 2010).

Based on these annotated examples, we can train a supervised classifier as proposed in the original paper. This constitutes a bootstrapping step. Through this step, we are able to make use of the complete set of features that we can extract from the classifier, promising further improvements over the distantly annotated results.

Transfer Learning

Transfer learning (TL, Thrun, 1996) is concerned with making use of knowledge gained from learning a model for a certain task when faced with another task that is related but different. Most of the time, we will not be able to straightforwardly apply a learned model for one task to another task without encountering significant problems. To make knowledge transfer between tasks easier, Thrun proposes the following possible techniques:

Memory-based learning Transductive classification functions that work directly on the input examples rather than on an abstract representation of their properties in a learned model are thought to be beneficial for transfer learning. This is motivated by the fact that in these approaches, examples can be weighted against each other dynamically, e.g., by using a distance measure that relates examples from the two tasks.

Better distance measurements Directly related to this, Thrun argues that improving the distance measure may lead to improved transfer. He suggests that a distance function could be learned automatically.

Adapting representations Orthogonally to the distance function, the feature representations of the two tasks may be addressed. Thrun proposes to modify feature spaces in order to facilitate knowledge transfer. Although the original proposal intends to automatically learn a feature space mapping – an idea related to representation learning (cf. Bengio et al., 2013) – a well-selected feature set may be sufficient.

In this thesis, we focus on improving feature representations for transfer learning.

2.4 Discriminative Models

Discriminative models work by modeling the posterior probability $p(y|x)$ of an output y given the example x directly (Bishop, 2006). This distinguishes them from generative models which consist of a process of probabilistic steps, the generative story, that together form the posterior (effectively modeling the joint probability $p(y, x)$). Discriminative models enjoy popularity in NLP and other fields as they are more flexible than purely generative models and often yield superior performance. In the following sections, we will present two prominent classes of models, the maximum entropy model and the Neural Network model.

2.4.1 Maximum Entropy Model

The *maximum entropy* model is particularly popular in NLP (Berger et al., 1996). It is in part motivated through issues regarding feature extraction from which many NLP tasks suffer. Most feature functions produce features that are not statistically independent. For example, the extraction of both unigrams and bigrams leads to dependent features as the occurrence of a word in a unigram implies that it is also part of a bigram. More generally, the words in a text cannot be considered statistically independent as they were produced in sequence with specific semantics in mind. Therefore, word-based features extracted from a text will be correlated.

It is important that this property is taken into consideration in a statistical model. Some models, such as Naive Bayes, assume that all features are statistically independent. The maximum entropy (MaxEnt) model does not make this assumption. It is based on the maximum entropy principle which states that one should use the model which is closest to a uniform distribution unless contrary information is available. This causes the so-called *explaining away* effect (Wellman and Henrion, 1993), meaning that only those features which explain the training data best receive high weights. We follow Nigam et al. (1999) and Mount (2011) for the introduction of MaxEnt models.

Formally, the MaxEnt model maximizes the entropy $H(p)$ of the distribution $p(c|x)$:

$$H(p) = -\sum_{c \in \mathcal{C}} \sum_{x \in \mathcal{X}} p(c|x) \log p(c|x)$$

Further, we add constraints that (i) $p(c|x)$ needs to be maximal for the correct label, and (ii) that p needs to be a probability distribution. Formulating the Lagrangian and finding an optimum, we obtain the following model:

$$p(c|x) = \frac{\exp(\sum_{i \in \mathcal{F}} w_f^{(c)} \phi_i(x))}{\sum_{c' \in \mathcal{C}} \exp(\sum_{i \in \mathcal{F}} w_f^{(c')} \phi_i(x))}$$

This exponential model function called *softmax* in the context of neural networks. We will discuss it in more detail in the following section. The feature weights $\mathbf{w}^{(c)}$ can be learned by maximizing the model's log-likelihood $\mathrm{LL}(\mathbf{w}|D)$,

$$\mathrm{LL}_{\mathbf{w}}(\mathcal{D}) = \log \prod_{\hat{x}, \hat{y} \in \mathcal{D}} p(\hat{y}|\hat{x}),$$

for the data on a set of training examples through numerical optimization, applying for example gradient-based or quasi-newton methods. The log-likelihood function is related to the cross-entropy function which is used for neural network training. Indeed, the MaxEnt model can itself be interpreted as a neural network.

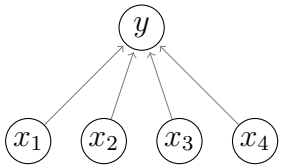


Figure 2.1: Artificial neuron

2.4.2 Neural Networks

Introduction

Neural networks (NN) are a class of mathematical model inspired by biology. The basic unit of a neural network is the artificial *neuron*, which is a model of a natural neuron that occurs in the brain.

The following introduction in part follows Bishop (2006). Formally, a neuron is a function that takes a vector $\mathbf{x}$ of input variables and returns an output $y = f(\mathbf{x})$, where f is the *output function* whose result y is the *activation* of the neuron. Neurons in neural networks are usually visualized as graphs. Figure 2.1 shows the graph for a single neuron. The nodes in the network are called *units*. The input units $x_1, \ldots, x_4$ are shown at the bottom, the output unit y at the top. n edge indicates that the value of the origin unit is used to compute the value of the target unit. A neural network consists of a set of neurons that are combined in some way, for example by composition of their output functions.

One of the most simple neural networks is the perceptron (Rosenblatt, 1958). It consists of a single neuron, like the one shown in Figure 2.1, that makes a classification decision based on a linear activation function of the inputs $\mathbf{x}$, a weight vector $\mathbf{w}$, and a bias term b:

$$y = f(\mathbf{x}) = \begin{cases} 1 & \text{if } \mathbf{w} \cdot \mathbf{x} + b > 0 \\ 0 & \text{else} \end{cases}$$

The weights and the bias are parameters that can be optimized numerically based on training data. The perceptron is still popular today in NLP, particularly for applications with complicated learning problems such as structured prediction where

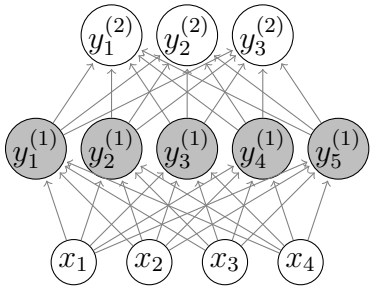


Figure 2.2: Multi-layer neural network. Hidden nodes are shaded in gray.

efficient training is key (Smith, 2011). The model led to the general form of a neuron with the output function

$$f(\mathbf{x}) = h(\mathbf{w} \cdot \mathbf{x} + b),$$

where we first compute the linear combination of the inputs $\mathbf{x}$ with the weights $\mathbf{w}$ and bias b and then apply an *activation function* h, which may be non-linear.

Logistic regression is an instance of this model, consisting of a single neuron with the logistic sigmoid $h(x) = \text{sigmoid}(x) = \frac{1}{1+e^{-x}}$ as its activation function. Logistic regression is closely related to the previously introduced maximum entropy model (cf. Mount, 2011).

As mentioned earlier, multiple neurons may be composed to form a complex network structure. If multiple neurons work on the same input units, each yielding a different outputs y_i, we call this structure a *layer*. To make subsequent notation easier, we can for each layer collect the weights $\mathbf{W} = [\mathbf{w}_1; \ldots; \mathbf{w}_n]$, the biases $\mathbf{b} = (b_1, \ldots, b_n)$, and the outputs $\mathbf{y} = (y_1, \ldots, y_n) = f(\mathbf{x})$.

Multiple layers can be combined through the composition of their output functions. For n layers with output functions $f^{(1)}, \ldots, f^{(n)}$, the final network output is

$$\mathbf{y} = (f^{(n)} \circ \ldots \circ f^{(1)})(\mathbf{x}) = f^{(n)}(f^{(n-1)}(\ldots(f^{(1)}(\mathbf{x})))).$$

This type of network setup is called *feedforward* as we can pass information only in the direction towards the output. There is no generally accepted convention

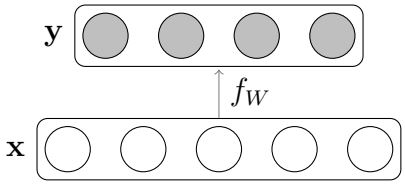


Figure 2.3: Compact neural network notation

for numbering layers. We adopt the convention where the input units are not counted as a layer. This has the advantage that the number of layers is equal to the number of parameter sets that need to be learned. An example graph for a multi-layer NN is shown in Figure 2.2. The network has two layers: an output layer (with three units $y_1^{(2)}, y_2^{(2)}, y_3^{(2)}$) which makes a prediction; and a *hidden* layer (with five units $y_1^{(1)}, \ldots, y_5^{(1)}$) which consists of network-internal values that are not specified in the training data and thus cannot optimized directly. In this model, calculating derivatives in the hidden layers is not straightforward as we can measure errors directly only at the output. Therefore, errors made at the output layer must recursively influence the weights of preceding layers as well. *Backpropagation* (Rummelhart et al., 1986) is an efficient algorithm for calculating the derivatives of a multi-layer network, making use of the chain rule for derivatives efficiently. The algorithm computes the derivatives iteratively, starting at the output layer, and uses the errors of each layer to compute of the derivatives in the preceding layer. For details of the algorithm, refer to (Bishop, 2006). The multi-layer perceptron is an instance of this idea in which each of the activation functions $h^{(1)}, \ldots, h^{(n)}$ is again the logistic sigmoid.

Vector Graph Notation

To make graph notation more readable and compact, we will employ a vector-based layout, which has been used previously for example by Bengio et al. (2003). Each layer is represented as a single node representing a vector, and a connection between the nodes indicates that the layers are fully connected. An example of this notation is shown in Figure 2.3. The graph shows two vectors with unit values $\mathbf{x}$ and $\mathbf{y}$, respectively, that are connected through a NN layer with output function

Name	Function	Range
identity	$\text{identity}(x) = x$	$[-\infty, \infty]$
hyperbolic tangent	$\tanh(x) = \frac{e^x + e^{-x}}{e^x - e^{-x}}$	$[-1, 1]$
logistic sigmoid	$\text{sigmoid}(x) = \frac{1}{1 + e^{-x}}$	$[0, 1]$
softmax	$\text{softmax}(\mathbf{x}, i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$	$[0, 1]$

Table 2.1: Examples of activation functions along with their ranges.

f which is parameterized with W. Sometimes, it is useful to split up layers into several vector boxes to indicate that these vectors are being concatenated.

Activation Functions

The motivation for using an activation function is, among others, that the dependency between the input and the expected output is not necessary linear. Further, it may be desired that the outputs of the network are bounded, e.g., between $[0, 1]$, or even normalized (i.e., the sum over all output units is 1). Ideally, the activation function should be differentiable to make numerical optimization easier.

We will briefly introduce the activation functions that are relevant in this thesis. Their definitions are shown in Table 2.1. All of these functions are smooth, i.e., infinitely differentiable. We will next give some motivation for each of the functions.

Identity The most straightforward function is the identity function that simply passes through the output of the linear equation.

Hyperbolic tangent The hyperbolic tangent, a function with an s-shaped graph, ranges between -1 and 1. It is often considered when the outputs are expected to be centered around zero, e.g., for normally distributed data with zero mean.

Logistic sigmoid The logistic sigmoid function, which again has an s-shaped graph, returns a value between zero and one for each output unit, which

makes it suitable for predictions that are encoded as binary vectors, as they are found for example in co-occurrence vectors in information retrieval.

softmax The softmax function returns a value between zero and one for each unit; the sum over all units is 1. Thus, softmax is particularly useful when the activations of all units are to be interpreted jointly as a probability distribution. The computation of softmax differs markedly from the other functions as it is the only one where the activation of one unit depends directly on the activations of all other units through normalization. The function that returns the a vector with softmax activations for all units in a layer is simply written $\mathbf{y} = \text{softmax}(\mathbf{x})$. Thus, the result over all training examples will be a matrix $\mathbf{Y}$ whose rows represent the examples and whose columns represent the individual classes.

Loss functions

The optimization objective for supervised neural network training is based on a *loss function* that measures how well the network performs on the training data. As the name implied, the output of the loss function needs to be minimized to receive the best-performing model (as opposed to maximization of the likelihood of the data under the model as it is common for MaxEnt). There are multiple loss functions which are common in the literature, and which are usually preferably used with specific output activation functions. We will briefly introduce the two most common functions. In the following, let $\hat{y}$ be the target value and y the predicted output. N denotes the number of examples, D the output dimensionality, i.e., the number of units.

Mean squared error (MSE) The MSE function assumes that the errors are normally distributed. It is defined as

$$MSE(\hat{\mathbf{y}}, \mathbf{y}) = \frac{1}{N} \sum_{n=1}^{N} (\hat{y}_i - y_i)^2.$$

MSE assumes that the errors are normally distributed. It is frequently used in combination with the tanh and linear activation functions.

Cross-entropy error (CEE) The CEE function measures the cross-entropy of the output and the target values. As a consequence, it is only applicable if the predictions and targets are probabilities. It is a natural choice for sigmoid and softmax activation functions. When used in combination with one of these functions, the computation of the derivatives in backpropagation can be simplified significantly. The CEE function for the multiclass case, i.e., when using softmax activation, is defined as follows:

$$CEE(\hat{\mathbf{Y}}, \mathbf{Y}) = \sum_{n=1}^{N} \sum_{d=1}^{D} \hat{y}_{nd} \ln y_{nd}$$

Optimization

Although research on optimization is not within the scope of this thesis, we briefly want to mention some points we consider important. Most importantly, neural network objective functions are in general not guaranteed to be convex. Thus, if we search for the minimum of such a function using convex optimization, we will find only local optima. While local optima can still be good enough to do well on a task, the result depends highly on initialization – which has consequentially been the focus of much research. Often, convex optimization methods (such as quasi-newton optimization (e.g., L-BFGS) or stochastic gradient descent) are applied to non-convex objectives, with the hope that the local optimum is sufficient for good performance.

Autoencoders

The *autoencoder* is a neural network model which has received much attention in recent research. It takes an input and tries to reproduce it at the output. This is accomplished by using an optimization objective that minimizes a loss function measuring the difference between the input and the output. The typical autoencoder setup is shown in Figure 2.4. It consists of two layers, a hidden layer, called the *encoding layer* (enc), and an output layer, called the *decoding* or *reconstruction*

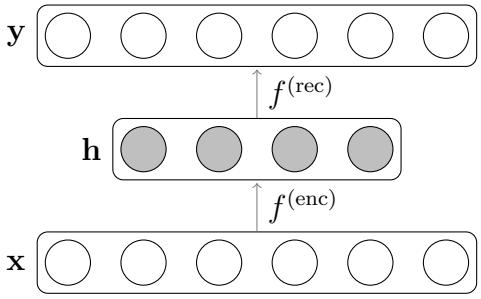


Figure 2.4: Autoencoder

layer (rec):

$$\mathbf{y} = f^{(\text{rec})}(f^{(\text{enc})}(\mathbf{x})) = h^{(\text{rec})}(\mathbf{W}^{(\text{rec})}h^{(\text{enc})}(\mathbf{W}^{(\text{enc})}\mathbf{x} + \mathbf{b}^{(\text{enc})}) + \mathbf{b}^{(\text{rec})})$$

The hope is that the hidden units learn to generalize from the input, identifying interesting correlations between the input features. The values of the hidden layer can be interpreted as a compressed representation of the input. In order to be able to reproduce the input, the number of output units must equal the number of input units. The number of hidden units is often chosen to be smaller than the number of input units to avoid that the network simply copies the input, the goal being to reduce the dimensionality of the data. Nonlinear activation functions can help to prevent copying as well. Depending on the way the data is encoded, certain combinations activation functions and loss functions are more beneficial than others (e.g., sigmoid and CEE for binary input vectors).

The idea of finding a representation with reduced dimensionality is not unlike the motivation for applying methods such as principal component analysis or singular value decomposition. In fact, they are closely related to autoencoders. Bourlard and Kamp (1988) for example show that singular value decomposition is equivalent to a two-layer feedforward network with linear activation and a mean squared error objective.

Deep Neural Networks

Deep neural networks (DNNs), i.e., networks with a high number of layers, are desirable from a learning point of view. It has been shown that neural networks are *universal approximators* (McCulloch and Pitts, 1943; Hornik, 1991), i.e., roughly spoken, they can approximate any continuous function. DNNs are advantageous to this cause as they can reduce the learning effort required for good approximation (Bengio, 2009).

For a long time, training arbitrarily deep neural networks has been considered too difficult for most applications. Combining multiple layers with non-linear activation functions makes the resulting objective function non-convex, which greatly complicates optimization. In some cases, for example on certain image processing tasks (LeCun et al., 1998), convex optimization finds useful local optima for DNNs, but generally, DNNs were unsuccessful.

Hinton and Salakhutdinov (2006) introduced a novel technique for learning deep networks. The key idea is to first *pre-train* each layer in the DNN in an autoencoder setup to initialize the weights for subsequent training of the full model. This effect is said to be related to non-convexity. The authors argue that pre-training serves to initialize the model weights beneficially so that convex optimization finds a better local optimum. These results have lead to much subsequent research where DNNs were shown to outperform state-of-the-art models in areas like speech recognition (Dahl et al., 2012) and image processing (Krizhevsky et al., 2012). Note that these are all cases where the data is easy to represent numerically. Transferring this success to text-based NLP remains a challenge.

Continuous Vector Space Language Models

The structure of language makes the application of DNNs for NLP difficult. DNNs originated in image processing where the neighborhood of two entities (pixels) has a straightforward interpretation due to their spatial relation. In NLP, neighborhood relations are not always interpretable easily and don't necessarily translate into a semantic relationship. Often, long distance dependencies hinder the use of locally sensitive models. Further, the basic units in image processing are numerical, as opposed to categorial word tokens found in text.

This problem motivates the use of *continuous vector space models*. Here, words are embedded in a continuous vector space, i.e., each word is represented by a continuous vector that is automatically learned. Usually, these spaces are low-dimensional compared to the high-dimensional sparse feature spaces that are usually used in NLP. One of the first models to learn word representations is the neural language model (NLM, Bengio et al., 2003). We will briefly introduce this model as it is the foundation on which the compositional models used in this thesis build.

For notation purposes, we first assume that we represent each word type as an integer $w \in \mathbb{N}$. We let $\mathbf{L}$ be the representation matrix where each row contains the vector for a certain word, and we define a lookup operation $\mathbf{l}_i$ on it that returns the i^{th} row of $\mathbf{L}$.

In a sense, the idea behind the NLM is similar to that of the autoencoder: given a set of context words, we want to be able to reconstruct the target word. In contrast to the autoencoder, the NLM sets up the problem as making a context prediction. The model takes a string of words as its input. Given a target word w_i, we take a windowed context of k words to the left and to the right of the word: $(w_{i-k}, \ldots, w_{i-1}, w_{i+1}, \ldots, w_{i+k})$. The input to the network will be the concatenation of the words' vectors, $\mathbf{x} = [\mathbf{l}_{w_{i-k}}, \ldots, \mathbf{l}_{w_{i+k}}]$. The network setup is shown in Figure 2.5: a two-layer network architecture where the first layer uses a tanh nonlinearity and the second layer uses softmax. All units except the output units are hidden. The network output will be a prediction of the identity of the target word w_i (i.e., a probability distribution indicating it):

$$\mathbf{y} = \text{softmax}(\mathbf{W}^{(2)} \tanh(\mathbf{W}^{(1)}\mathbf{x}^\intercal + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)})$$

The NLM is trained by learning both the usual weight parameters as well as the matrix $\mathbf{L}$, the input representations of the words. To improve the quality of the resulting word representations, randomly constructed negative examples have been shown to be helpful (Collobert and Weston, 2008). Sometimes, skip connections are used to connect the input layer directly to the prediction layer (Bengio et al., 2003).

The purpose of learning word representations is two-fold. First, the representa-

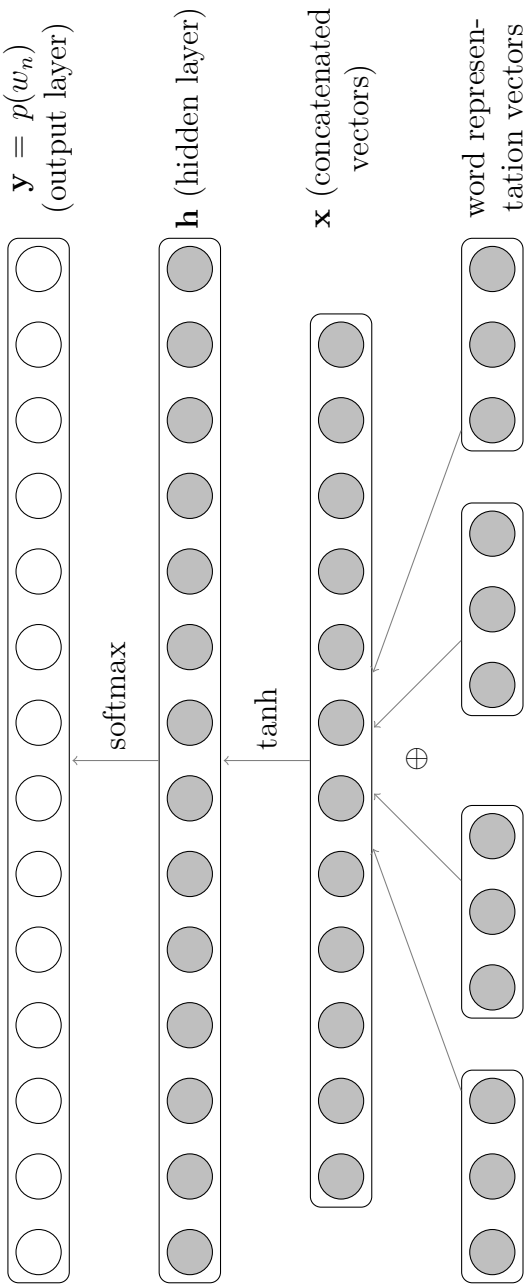


Figure 2.5: Neural language model

tions may be used in any task where vector representations of words are required, e.g., in information retrieval. Second, the representations serve as pre-training for subsequent hierarchical neural network models. We will present and analyze such a model in Chapter 7.

2.4.3 Regularization

An important goal in supervised learning is to find a model that generalizes well, meaning that it works well on both the test and training data. Models that capture the training data well but not the test data are said to *overfit* the training data. It is thus desirable to take measures to avoid overfitting. For this reason, it has become best practice to *regularize* discriminative models. As we can freely manipulate weights (as opposed to, e.g., probabilistic generative models where the parameters are required to be probabilities), we can simply impose a constraint on them. A common method for regularization restricts the growth of the model parameters $\mathbf{w}$, which prevents overfitting. To accomplish this, the original optimization objective $E_{\mathbf{w}}(\bullet)$ is modified to jointly minimize a norm of $\mathbf{w}$, for example its L_2 norm $\|\mathbf{w}\|_2 = \sqrt{\sum_i |w_i|^2}$:

$$E_{\mathbf{w}}^{(\text{reg})}(\bullet) = E_{\mathbf{w}}(\bullet) + \frac{\|\mathbf{w}\|_2^2}{2\sigma^2}$$

Here, σ is a parameter controlling the degree of regularization. This regularization term corresponds to a Gaussian prior on the weight vector where σ^2 is the variance. All discriminative models used in this thesis are regularized.

2.5 Graph-Based Natural Language Processing

Recently, graph-based representations of natural language have emerged as a powerful alternative to purely inductive learning methods (Mihalcea and Radev, 2011). One idea which has strongly influenced graph-based approaches to NLP is *transduction*, where knowledge from the training data is directly transferred to the test data instead of training a mathematical model. This idea is naturally applicable on graphs as they encode relations between objects. Often, properties such as transitivity may be exploited.

We will next give a brief introduction to graph theory, based on the text by Diestel (2000), focusing on the concepts that find application in this thesis. We then discuss ways to represent natural language through graphs. Finally, we introduce the graph algorithms that we apply.

2.5.1 Graph Theory

We formalize *directed graphs* as $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, where $\mathcal{V}$ is the set of *nodes* v, $\mathcal{E}$ the set of *directed edges* $\langle u, v \rangle$ between two nodes u and v, and $\mathbf{W}$ a matrix of *edge weights* w_{uv} between u and v. We next describe some basic concepts of graph theory. Note that all definitions can easily be modified for *undirected graphs* by assuming unordered pairs (u, v) as edges and a symmetric $\mathbf{W}$.

Degree The *out-degree* $g_{out}(i)$ of a node i is the number of edges that start at i, i.e., $g(i) = |\{(i,j)|j \in \mathcal{V} \wedge (i,j) \in \mathcal{E}\}|$. The *in-degree* g_{in} is defined analogously for edges that end at i.

Distance The distance $d(i,j)$ of two nodes i and j is the length of the shortest path between them; $d(i,i) = 0$ and if there is no path between i and j, $d(i,j) = \infty$.

Subgraph $H = (\mathcal{V}', \mathcal{E}', \mathbf{W}')$ is a subgraph of a graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$, written as $H \subseteq G$, iff $\mathcal{V}' \subseteq \mathcal{V}$ and $\mathcal{E}' \subseteq \mathcal{E}$ and $\mathbf{W}'$ is the appropriate submatrix of $\mathbf{W}$.

Connected graph A graph (or subgraph) is called connected iff there is a path between any pair of its nodes.

Component A subgraph H of G is a component of G iff there is no H', $H \subseteq H' \subseteq G$, which is also connected.

Neighborhood The k-neighborhood of a node i is the subgraph $N = (\mathcal{V}', \mathcal{E}', \mathbf{W}')$ of $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ where $\mathcal{V}' = \{j|\, d(i,j) \leq k \wedge j \in \mathcal{V}\}$ and $\mathcal{E}' = \{\langle i,j \rangle|\, \langle i,j \rangle \in \mathcal{E}\} \cup \{\langle j,i \rangle|\, \langle j,i \rangle \in \mathcal{E}\}$ and $\mathbf{W}'$ is the appropriate submatrix of $\mathbf{W}$.

2.5.2 Graph Representations of Natural Language

There are many ways of representing language in this formalism for different NLP problems (based on Mihalcea and Radev, 2011, Chapter 4). We will briefly introduce the setups that we apply in this thesis.

Co-occurrence graphs model co-occurrence relations between objects. In the most basic setup, the *word graph*, each node represents a word type. Undirected edges between words represent co-occurrence of the words in a text corpus. One motivation behind this graph structure is that when choosing the right relations, the graph might implicitly encode semantic information, such as semantic similarity between two words. The graph is constructed based on some text by taking a window of n words to the left and right of a word w_x and add an undirected edge (w_x, w_{x+i}) to the graph for each $-n \leq i \leq n | i \neq x$. The edges are weighted by the overall co-occurrence count of the word pair. If necessary be, this graph can be made directed by splitting each edge into an ingoing edge $\langle u, v \rangle$ and outgoing edge $\langle v, u \rangle$. This approach has several drawbacks. First, it is noisy as not all words in the neighborhood have a strong relationship. Second, it leads to a large, dense graph which may be hard to process.

To address these issues, a more conservative approach has been proposed, leading to a *syntactic dependency graph*. Related words are extracted based on pre-defined syntactic patterns, such as verb-object or conjunction relations. If two words are found in such a relation in a corpus, they are assumed to be co-occurring and an edge is added to the graph as described above.

A further extension to this framework is to include edges that do not strictly model co-occurrence but rather occurrence. Due to a lack of a term in the literature for this structure, we call it *occurrence network* analogously to the previously introduced concepts. For example, we can introduce a node representing a document d and create edges (d, w) between d and each word $w \in d$.

To model discourse phenomena in language, *adjacency graphs* have been proposed. Adjacency graphs are a stricter form of co-occurrence graphs where the edges between two nodes are weighted by how close the objects they represent are to each other. This can be accomplished by defining a weight function based on proximity or even a strict cutoff after a number of neighbors. This way, the

resulting graph will have chain-like structures representing sequence information about linguistic entities.

An interesting but frequently overlooked distinction is whether the nodes in the graph represent *types* or *tokens*. Word graphs usually model word types in order to achieve semantic generalization. Discourse graphs often use tokens to model objects in context, for example to identify informative sentences for summarization. Sometimes, a hybrid approach may desired, for example for word sense disambiguation.

2.5.3 Graph Algorithms

Most popular graph-based machine learning algorithms are designed for transductive learning. This means that they do not perform parameter-based training of mathematical models. Instead, they try to make use of the training information directly, for example by propagation, segmentation, or the computation of node-node relatedness. This also emphasizes the importance of choosing a suitable graph structure. We will introduce two graph algorithms, PageRank and minimum cut.

PageRank

PageRank is an algorithm for measuring how well-connected each node in a graph is (cf. Bianchini et al., 2005). It was introduced by Page et al. (1999) to rank web pages by user preference in order to improve the results of their web search engine. They accomplished this by viewing the web as a graph where web sites are nodes and hyperlinks between the pages define edges between the nodes. The underlying concept of PageRank is that of a random walk, which was described initially by Pearson (1905). We formalize random walks based on the introduction by Abney (2010). A Markov chain is a stochastic process that models the transition between a set of states $\mathcal{V}$. It is defined through the transition probabilities $p_{ij} \in \mathbf{P}$ for all states i and j. States and transitions between them can be represented as a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{P})$ with $\mathbf{P}$ as the row-normalized edge weight matrix, where $\mathcal{E} = \{(i, j)|p_{ij} \neq 0\}$.

An important property of a Markov chain is egodicty. A Markov chain is ergodic iff it is irreducible and aperiodic. Irreducibility is given if there is a path

with non-zero probability between any two states. Aperiodicity requires that there is no periodic path between any two states. Any ergodic Markov chain has a unique steady state probability distribution. This distribution is the dominant left eigenvector of the transition probability matrix $\mathbf{P}$. We refer to the dominant left eigenvector as the rank vector $\mathbf{r}$. It can be computed as the fixed point of the following equation, e.g., by the power method:

$$\mathbf{r} = \mathbf{r}\,\mathbf{P} \tag{2.1}$$

Ergodicity is difficult to ensure in practical applications as we generally have little influence on the structure of the graphs that arise. To avoid the issue on the web graph, the transition matrix of the graph can be altered to pairwisely give all nodes a non-zero transition probability through interpolation with another non-zero probability matrix, a process known as *teleportation* (cf. Haveliwala, 2003). For web data, teleportation could be interpreted as the surfer manually moving to a different web site rather than following a link. Formally, teleportation can be introduced into an existing Markov chain as follows. Assume that we are given an $N \times N$ transition matrix $\mathbf{A}$ for a Markov chain, an N-dimensional vector of target teleportation probabilities $\mathbf{t}$, and an overall teleportation probability $(1-\alpha)$ which specifies. We can then compute the ergodic probability matrix $\mathbf{P}$ through interpolation as

$$\mathbf{P} = \alpha\mathbf{A} + (1-\alpha)\mathbf{1}^\intercal\mathbf{t}. \tag{2.2}$$

In standard PageRank, $\mathbf{t}$ is the uniform distribution: all nodes are equally likely to be teleportation targets.

Personalized PageRank PageRank measures the overall connectedness of a node in the graph. In many applications, however, it is of interest to measure the connectedness with respect to a set of relevant nodes which we call *seed nodes*. One algorithm that solves this problem is *Personalized PageRank*, also called *Topic-Specific PageRank*. It was originally intended to compute the PageRank of web pages with respect to a certain topic or focus of interest.

Personalized PageRank is computed with respect to a set of seed nodes $\mathcal{S}$ that represent known instances of relevance. Based on this set, we will bias PageRank to

give higher weight to nodes that are well-connected to these nodes and therefore are good representatives of the topic that dominates $\mathcal{S}$. We accomplish this by setting the teleportation probabilities t_i in $\mathbf{t}$ non-uniformly based on whether a node n_i is in $\mathcal{S}$:

$$t_i = \begin{cases} \frac{1}{|\mathcal{S}|} & \text{if } n_i \in \mathcal{S} \\ 0 & \text{else} \end{cases} \tag{2.3}$$

Thus, teleportation will occur only to nodes in $\mathcal{S}$. This causes $\mathbf{r}$ to have high values for nodes that are well-connected to nodes in $\mathcal{S}$.

Minimum Cut

The *minimum cut* (MinCut) problem is the problem of finding a segmentation of a graph into two parts where the sum of the weights of the edges that have to be removed is minimal. We will formalize the problem based on (Abney, 2010) and briefly discuss algorithms for computing the MinCut.

We write the complement of a set of nodes S as $\bar{S}$. Each set of nodes S has a boundary ∂S which is the set of all edges that connect nodes in S to nodes in $\bar{S}$. The size of this boundary can be computed as the sum of the weights of ∂S, and is also refereed to as a *cut*:

$$size(\partial S) = cut(S, \bar{S}) = \sum_{\langle a,b \rangle \in S \times \bar{S}} w_{ab} + w_{ba}$$

The problem of finding the cut with the smallest boundary size is the minimum cut problem. $size(\partial S)$ is the value of the cut. The result is a partitioning of the nodes of the graph into two sets S and $\bar{S}$ with optimal similarity within each set and optimal dissimilarity between the sets.

Much research covers the problem of finding the minimum cut. In this thesis, we will make use of a special case of MinCut, the *s-t-MinCut* where two nodes, s and t, are required to be on different sides of the cut, i.e. $s \in S$ and $t \in \bar{S}$. Many algorithms, such as push-relabel algorithms (Cherkassky and Goldberg, 1995), exploit the equivalence of the s-t-MinCut problem to the maximum flow (MaxFlow) problem for flow networks. In this paradigm, the graph is viewed as a flow network. Edge weights correspond to the capacity of flow between two nodes.

Edges must be directed and there must be one node each representing the source, where flow can enter the network, and the sink, where the flow leaves the network. The goal of the MaxFlow problem is to find the maximum flow value from the source s to the sink t, which is equal to the value of the cut. When using blocking algorithms such as the push-relabel algorithm (Cherkassky and Goldberg, 1995), the graph is partitioned by the "frontier" of blocked edges.

2.6 Evaluation

2.6.1 Evaluation Measures

For the evaluation of classification results, we resort to well-known measures from information retrieval (Manning et al., 2008). All evaluation measures presented in the following section rely on some basic counts on a test collection of data $\mathcal{D}$.

The basic measurements are the counts of true positives (tp), true negatives (tn), false positives (fp) and false negatives (fn) with respect to each class c of each example. These depend on whether the class predicted by the classifier matches the expected prediction, i.e. the *true* class, as shown in Table 2.2.

		true	
predicted		c	$\neg c$
	c	$tp^{(c)}$	$fp^{(c)}$
	$\neg c$	$fn^{(c)}$	$tn^{(c)}$

Table 2.2: Confusion matrix of true and predicted class

We define $tp^{(c)}$, $tn^{(c)}$, $fp^{(c)}$, and $fn^{(c)}$ to denote the number of times the respective events occurred in the collection for a class c. Based on these count statistics, we define our evaluation measures.

The most basic measure is *accuracy* (Acc). Here, we simply measure the ratio of correctly classified examples on the collection:

$$Acc = \frac{\sum_{c \in \mathcal{C}} tp^{(c)}}{|\mathcal{D}|}$$

Accuracy is a good measure when classes are distributed uniformly in the collection. However, as class imbalances grow more pronounced, high accuracy might be attained by a classifier that has a bias towards the majority class.

Precision and recall are often used as an alternative, providing a more detailed analysis of the classifier's behavior with respect to each class c. *Precision* ($P^{(c)}$) measures the relative frequency of correctly classified examples that were predicted to belong to c:

$$P^{(c)} = \frac{tp^{(c)}}{tp^{(c)} + fp^{(c)}}$$

Conversely, *recall* ($R^{(c)}$) measures the relative frequency of correctly classified examples among the set of examples whose correct class is c:

$$R^{(c)} = \frac{tp^{(c)}}{tp^{(c)} + fn^{(c)}}$$

The harmonic mean of precision and recall is called the F *measure*. In this thesis, we use the balanced F measure, or F_1 measure, i.e. precision and recall are weighted equally:

$$F_1^{(c)} = \frac{2P^{(c)}R^{(c)}}{P^{(c)} + R^{(c)}}$$

We use the *macro-averaged* F measure (abbreviated as $\bar{F}_1$) over all classes as a measure for overall classifier performance:

$$\bar{F}_1 = \frac{\sum_{c \in \mathcal{C}} F_1^{(c)}}{|\mathcal{C}|}$$

The F measure of each class contributes equally to the macro average. This way, class-imbalances in the data do not have an influence on the calculation of the result.

2.6.2 Hypothesis Tests

When developing and evaluating natural language processing systems, it is important to know whether using one method instead of another improves the results. For this reason, we perform the aforementioned evaluation steps. However, from a

Algorithm 1 Approximate randomization test

Input: $\mathbf{y}_1$, $\mathbf{y}_2$, $\hat{\mathbf{y}}$ k
Ensure: $|\mathbf{y}_1| = |\mathbf{y}_2|$
$m_0 \leftarrow \mathit{eval}(\mathbf{y}_1, \hat{\mathbf{y}}) - \mathit{eval}(\mathbf{y}_2, \hat{\mathbf{y}})$
$s \leftarrow 0$
for $i = 1$ to k **do**
  $\mathbf{y}_1' \leftarrow \mathbf{y}_1$, $\mathbf{y}_2' \leftarrow \mathbf{y}_2$
  for j = 1 to $|\mathbf{y}_1|$ **do**
    if Bernoulli(0.5) = 1 **then**
      $\mathit{swap}(\mathbf{y}_1'[j], \mathbf{y}_2'[j])$
    end if
  end for
  $m \leftarrow \mathit{eval}(\mathbf{y}_1', \hat{\mathbf{y}}) - \mathit{eval}(\mathbf{y}_2', \hat{\mathbf{y}})$
  if $m \geq m_0$ **then**
    $s \leftarrow s + 1$
  end if
end for
return s/k

statistical point of view, it could still be possible that one method outperformed the other by chance. To check this, we can make use of statistical *hypothesis tests*. We briefly introduce the general problem based on Yeh (2000).

Assume that we have two systems which produced the outputs $\mathbf{y}_1$ and $\mathbf{y}_2$, which we evaluate according to some evaluation function *eval* (such as accuracy or F_1). We make the *null hypothesis* that there is no difference between the outputs of *eval* for $\mathbf{y}_1$ and $\mathbf{y}_2$. We then estimate the probability of the observed difference between $\mathbf{y}_1$ and $\mathbf{y}_2$ given the null hypothesis. If this probability is lower than a threshold – the *confidence level* – we reject the null hypothesis and say that the difference between the results is *statistically significant*.

There is a large variety of hypothesis tests available for the previously introduced evaluation measures. As noted by Yeh (2000), many of these tests assume statistical independence between the two systems. This assumption is mostly wrong in practice as many systems share components, for example by using the same core feature sets. Such tests underestimate statistical significance and therefore tend to

yield false negative results.

In this thesis, we adopt the *approximate randomization* test (Noreen, 1989; Yeh, 2000). The underlying idea is to test the null hypothesis through simulation. If there is indeed no difference between the two systems, the results they produce should be interchangeable. We can test this by permuting the predictions of the systems between each other. In total, this would lead to 2^n permutations for n examples. For each permutation, we check whether the difference between the systems according to our evaluation measure is as least as large as the one for the unpermuted systems. The relative frequency of how often this occurs will tell us how likely it is that we observed an improvement by chance. Since in practice we deal with large datasets, we generally cannot perform 2^n permutations. For this reason, we approximate the result by generating a sufficiently large number of permutations randomly.

The approximate randomization algorithm is shown in Algorithm 1. It has a free parameter k which determines the number of random samples. In addition, we need to specify an evaluation measure *eval*. Each sample is generated by randomly permuting the predictions for each test example between the tests. In each iteration, we test whether the difference m between the two results is larger than the initial difference m_0. The probability of accepting the null hypothesis is $p = s/k$, where s is the number of trials in which $m \geq m_0$.

This test has several advantages. First, it does not make the independence assumption described above. Second, it is applicable to any evaluation measure and does not need to be derived again if the measure is changed. On the other hand, as the name implies, the test is not exact but instead iteratively tests permutations of the results. The result becomes more reliable with increasing k. For this reason, it is more expensive to compute than an exact test.

Unless stated otherwise, we run the test with 10,000 iterations and use a confidence level of $p = 0.05$.

2.6.3 Agreement Measures

Annotation of linguistic properties by humans is an essential foundation of computational linguistics. Both supervised training and evaluation of NLP methods

require human annotations. Consequentially, assessing the quality of such annotations received much attention. In order to check whether human annotators work in a consistent and reproducible manner, one can directly compare the work of two or more annotators using an *agreement measure*. The following introduction is based on the surveys by Artstein and Poesio (2008) and Gwet (2008) .

Defining an agreement measure is not straightforward. For example, we could simply measure the relative frequency of times the annotators agree. This measure would, however, not take into account the fact that the distribution of answers is usually skewed. This is why measures of pure observed agreement do not yield a realistic estimate. To this end, it has been proposed to correct the observed agreement probability for chance agreement, which lead to the *kappa* measure (κ):

$$\kappa = \frac{P_a - P_e}{1 - P_e}$$

P_a is the probability of observed agreement, and P_e the probability agreement by chance. There is no general consensus regarding how chance agreement should be calculated, which is why there are many versions of κ. We measure agreement with Fleiss's κ. It has the advantage that it can be defined so that it is applicable to an arbitrary number of annotators. Assume N examples which can be annotated with K categories. Let A be the total number of annotators. Let a_{nk} be the number of annotators who annotated example n with category k. Then, the observed agreement P_a can be computed as follows for an arbitrary number of annotators:

$$P_a = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{\binom{A}{2}} \sum_{k=1}^{k} \binom{a_{nk}}{2}$$

Essentially, we calculate the agreement for each example n as the number of pairwisely agreeing annotators $\binom{a_{nk}}{2}$ normalized over all possible pairings. We then take the mean over all examples. For Fleiss's κ, chance agreement is defined based on a single distribution over all annotators. The probability for each category k is simply

$$p_k = \frac{1}{AN} \sum_{n=1}^{N} a_{nk}.$$

Chance agreement P_e can then be calculated as

$$P_e = \sum_{k=1}^{K} p_k^2.$$

2.7 Summary

In this chapter, we motivated and introduced statistical natural language processing. We showed how language can be represented for statistical processing and introduced a set of learning strategies. We presented several statistical models that are frequently applied in NLP. Finally, we described the means of evaluation we employ in this thesis.

3 Sentiment Analysis

Sentiment Analysis is an area of natural language processing that is concerned with capturing attitudes expressed in linguistic expressions. *Sentiment* describes an opinion or attitude expressed by an individual, the *opinion holder*, about an entity, the *target*. *Attitudes* – "relatively enduring, affectively colored beliefs, preferences, and predispositions towards objects or persons" (Scherer, 2003) – are different from *emotions* – "brief episodes of synchronized responses" (Scherer, 2003) as a reaction to external influences. This distinguishes sentiment analysis from other problems such as emotion analysis where the general emotional state (influenced by various external factors) is of interest, and not the attitude towards a specific target. The degree and direction of sentiment (i.e., how positive or negative it is) is called its *polarity*.

Research on the emotional potential of linguistic expressions has a long history in linguistics. An important related concept that has been introduced early is connotation, the additional emotional or cultural meaning of an expression (cf. Linke et al., 1994). Connotation is an important linguistic foundation for expressing sentiment. Computational approaches to sentiment emerged in the 1990s. It has been recognized early that identifying the author's attitude towards a topic is interesting, for example in information retrieval (Hearst, 1992). Research on fully automatic sentiment analysis goes back to Spertus (1997) who identifies hostile messages on the web with a combination of rules and a lexicon. The breakthrough of statistical models for large-scale sentiment analysis was introduced by Pang et al. (2002) who showed that user-supplied polarity ratings of product reviews can be predicted automatically with high accuracy through supervised classification. Following this research, numerous sentiment analysis approaches on a broad range of linguistic levels have been presented. These range from low-level units like words and phrases (Turney, 2002), to sentences (Pang and Lee, 2005) to fine-grained document analysis (Hu and Liu, 2004). Often, sentiment analysis is supported by large vocabulary lexical resources such as polarity lexicons (Wilson et al., 2005b). Recently, contextual properties have become increasingly important, reflecting the

fact that word senses (Wiebe and Mihalcea, 2006), domains (Mullen and Collier, 2004), and topics (Blitzer et al., 2007) influence the way sentiment is expressed.

An important research focus in sentiment analysis is *subjectivity analysis* (Wiebe, 1990), where the objective is to automatically determine whether a statement is subjective (i.e., reflecting an individual's inner state of mind) or objective (i.e., verifiable in reality).

Sentiment analysis also plays a role in practical applications and downstream tasks. Sentiment analysis systems may be components of information retrieval (Hearst, 1992) systems, or may be used to summarize the vast amount of opinionated content that has become available (e.g., Hu and Liu, 2004). Amazon.com, for example, recently started to present summarizing quotes from reviews, including a quantitative estimation of how often similar statements were found in other reviews. Sentiment analysis also has a longstanding history in automatic stock market analysis (e.g., Baker and Wurgler, 2007; He et al., 2013), where the automatic analysis of investor and customer sentiment can be used to predict stock price developments.

The rest of this chapter is structured as follows. We will first introduce basic concepts of sentiment analysis such as polarity and subjectivity in Section 3.1. In Section 3.2, we present approaches to the automatic classification and detection of sentiment on different linguistic levels. Finally, we discuss possible feature representations for sentiment analysis in Section 3.3.

3.1 Concepts in Sentiment Analysis

3.1.1 Polarity

The direction and strength of sentiment is called *polarity*. The simplest and most common polarity scheme assumes two categories, *positive* and *negative*. These two categories constitute the extreme ends of a discrete or continuous scale. This definition covers most voting schemes used in practice, such as

- thumbs up/down (e.g., Facebook, YouTube),
- positive, neutral, negative (e.g., eBay), or

- star ratings (e.g., Amazon, IMDb).

Often, polarity is mapped to the $[-1, 1]$ interval, assuming that -1 is the most negative polarity possible, and 1 the most positive one. There is some ambiguity regarding the center of the scale (0), which is commonly described as neutral. Note however that this can also mean a more or less balanced mix of positive and negative content (Koppel and Schler, 2006). It has been recognized that this data is difficult to assess even for humans (Kim and Hovy, 2004), which is why data from this category is sometimes omitted from experiments to simplify the problem (e.g., Blitzer et al., 2007).

Recently, more complex polarity problems, bordering emotion analysis, have been investigated. Socher et al. (2011), for example, propose the use of the scheme from the experience project[3] which collects user-submitted stories. Here, readers can vote for one of six tags – *you rock*, *tehee*, *I understand*, *sorry*, *hugs*, and *wow, just wow*, which cover a broader emotional spectrum. The result of these votings is a distribution over tags, which is significantly more difficult to model than previous schemes that simply require the prediction of discrete classes.

In this thesis, we are concerned with binary polarity prediction problems, thus focusing only on the two extreme points (*positive* and *negative*) of the polarity scale.

Prior and Contextual Polarity

Naturally, clues may be ambiguous, leading to different polarities in different contexts. The polarity of a clue out of context is called the *prior polarity*, the polarity in context is the *contextual polarity* (Wilson et al., 2005b). Contextual effects may arise for different reasons. Some clues are perceived differently depending on the target. For example, *funny* can express positive sentiment when used in the movie domain (*funny story*) or negative sentiment when used to describe food (*funny taste*). Sometimes, one can observe cultural effects as well, such as *warm beer* which triggers different reactions in different communities. The automatic recognition of contextual polarity has been attempted using various linguistic context features (Wilson et al., 2005b). However, it remains a challenge as domain and

[3] `http://www.experienceproject.com/`, which has since abolished this voting scheme

topic effects have a strong influence. Thus, in this thesis, we will investigate only prior polarity.

Polarity Clues and Polarity Lexicons

The most frequently investigated basic unit in sentiment analysis are words. Commonly, each word or multi-word unit is viewed as having a polarity which contributes to the sentiment of higher-order units. Interestingly, at the word level, sentiment analysis touches related fields such as emotion analysis, because the notion of expressing sentiment towards a target vanishes. This distinction is often ignored if the target is known. In this case, words and phrases are treated as *clues* for the expressed sentiment. We will refer to this approach as the *clue model*.

Polarity Lexicons are an important resource in sentiment analysis. A polarity lexicon is a collection of clue words or phrases annotated with polarity. Often, they contain additional information such as part of speech (e.g., Wilson et al., 2005b), word senses (e.g., Su and Markert, 2008) or domain information (e.g., Choi and Cardie, 2009). There are not many polarity lexicons available that have been fully manually annotated. Thus, much research has focused on automatically generating or extending lexicons by exploiting the semantic relatedness of clues (Scheible et al., 2010; Hassan and Radev, 2010).

Polarity lexicons may be used as a standalone resource or in combination with a statistical classifier, which is, however, not a trivial task; this problem is one of the main research questions addressed in this thesis.

3.1.2 Subjectivity

It has been recognized in philosophy that there is a distinction between subjective impressions of an individual and objective reality (Solomon, 1977). This distinction manifests itself in human communication as well, leading to *subjective and objective language* (Banfield, 1982). Statements can be subjective or objective depending on their verifiability (cf. Liu, 2012, Section 2.4):

Subjective A subjective statement expresses the inner state of mind of an individual. It is not directly verifiable. An example for a subjective statement is

the following sentence:[4]

(3.1) *One of the greatest romantic comedies of the past decade.*

This sentence expresses the viewer's subjective view of the movie, particularly due to the subjective expression *greatest*.

Objective An objective statement is factual. It describes observable reality and it is directly verifiable by individuals other than the one who made the statement. The following sentence is considered objective:

(3.2) *The movie won a Golden Globe for best foreign film and an Oscar.*

It is objectively verifiable that the movie in question has indeed won said awards.

The automatic classification of expressions into these categories is called *subjectivity analysis*.

It has been recognized that subjectivity is not an indicator for whether an expression bears sentiment or not (cf. Liu, 2010). Consider Example 3.3, which contains a factual statement.

(3.3) *I have to admit I walked out of Runteldat.*

The truth of this statement can be verified in reality (e.g., by observers who saw the author walk out), thus it is objective. Yet, the sentence expresses negative sentiment towards the movie that it describes as the author walked out of the movie because he did not enjoy it. This example illustrates that subjectivity and polarity are orthogonal. This notion is reflected in several annotation studies (e.g., Wiebe et al., 2005; Su and Markert, 2008) where the two concepts were treated accordingly

Nevertheless, subjectivity has been used as a criterion of how interesting an expression is for sentiment analysis. Pang and Lee (2005) for example propose subjectivity-based selection for summarizing reviews. Many researchers (e.g., Wilson et al., 2005a; Lin and He, 2009; Heerschop et al., 2011) use subjectivity analysis as a pre-processing step to filtering data for sentiment classification.

[4]All following examples are taken from the Pang & Lee sentence dataset.

Note also that the definition of subjectivity given above is not universally accepted. While we adopt the classical definition from philosophy as described above, many related concepts are often presented under the guise of subjectivity. Most notably, the approach by Pang and Lee (2005) uses proxy data in place of any formal definition of subjectivity, showing that practical concerns are often of higher importance than formal correctness.

3.2 Automatic Sentiment Analysis

Automatic sentiment analysis is concerned with predicting the polarity of unseen data. The term covers a broad range of problems and approaches. In this section, we will discuss three general types of sentiment analysis problems. There is a highly diverse set of methods tackling these problems are which cannot easily be summarized briefly. We will introduce the relevant methodology in more detail in the respective sections. In general, they can be divided into two major classes: rule-based and statistical approaches. In this thesis, we focus mainly on statistical approaches.

In today's research, many different views on automatic sentiment analysis exist, leading to different tasks. The most prominent difference between them is the granularity of analysis. Sentiment analysis has been performed on multiple linguistic levels. The dominant tasks are the prediction of polarity on the document, sentence/clause, and entity/aspect level. In this section, we will introduce these levels based on the overview by Liu (2010). In the experimental part of the thesis, we work on either the document or the sentence level.

3.2.1 Document Level

The goal of document-level sentiment analysis is to predict the overall polarity expressed in a document. Typically, the documents on which this type of analysis is performed are ones in which the author evaluates only a single entity, such as reviews of products, hotels, or movies.

Figure 3.1 shows three example documents. Figure 3.1a is a product review from Amazon.com in which a user/customer reviews a book. In addition to the

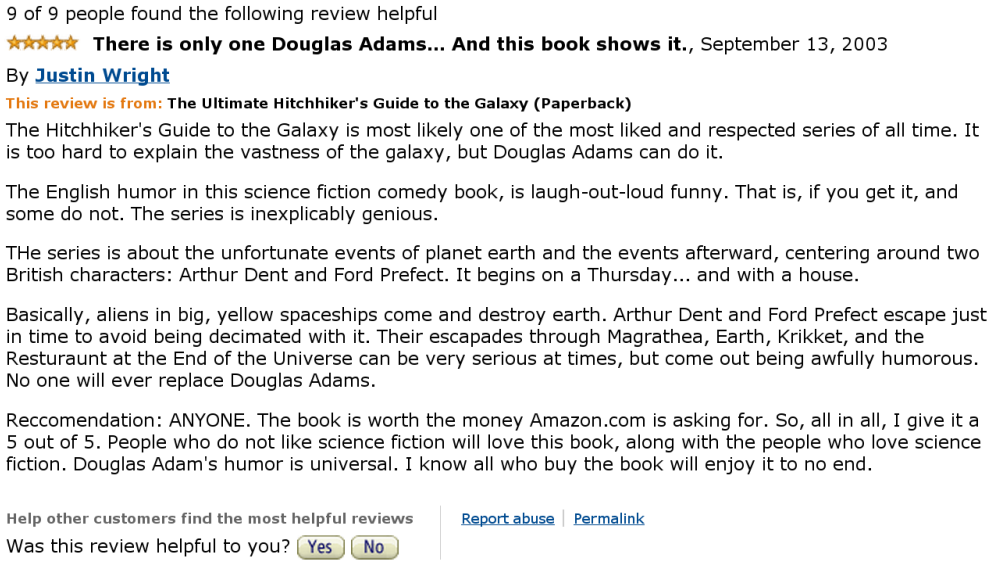
review text, the user supplies a rating on a categorial polarity scale, in this case spanning 1–5 stars. Figure 3.1b shows a hotel review from TripAdvisor, a travel website. Again, we see a user-supplied review as well as an overall rating which is supplemented by several ratings of certain *aspects* regarding their stay, such as the location of the hotel. The automatic prediction of aspects would in this case be a document-level prediction task (as addressed for example by Titov and McDonald, 2008). It is also possible to view it as an entity-level task (cf. Section 3.2.3). Figure 3.1c contains a movie review from the Internet Movie Database (IMDb). Again, the reviewer can rate the movie on a pre-defined scale of 1–10 stars. In comparison, we note that the this review is much longer than the other two. The average document length may vary between domains (Blitzer et al., 2007).

The task of predicting document-level polarity can be cast as a standard text classification problem. The problem can then be addressed using well-established techniques, such as maximum entropy classification (Pang et al., 2002) using word-based feature representations (cf. Section 3.3). There are several assumptions involved in the text classification approach. First, it is assumed that the whole text is concerned with a single target, namely the product that is the subject of the review. Second, the author is assumed to be the opinion holder. Third, the clue features are assumed to be noisy but overall predictive of the document-level polarity. These assumptions might not hold in general for arbitrary statements. In news texts, for example, opinions by different holders about various targets can be re-stated by the news reporter. There are also in exceptions seemingly unproblematic genres such as product reviews. In the following example,[5] the author makes a comparison to a different product, which might be wrongly recognized as negative sentiment towards the reviewed product:

(3.4) *You can get very cheap ones (around $5) that look like EON's, but they usually say it lasts 5 years, with 1-hour of continuous use. That definitely* ***looks bad in comparison to*** *this one, which claims a 168 hour use.*

As documents are typically long enough, violations of the three assumptions above may be disregarded as noise. Instead, we rely on the sufficient availability of unproblematic clues. Thus, the approach often fails for short documents which might

[5] Taken from `http://www.amazon.com/product-reviews/B000095RB7`


9 of 9 people found the following review helpful

★★★★★ **There is only one Douglas Adams... And this book shows it.**, September 13, 2003

By **Justin Wright**

This review is from: **The Ultimate Hitchhiker's Guide to the Galaxy (Paperback)**

The Hitchhiker's Guide to the Galaxy is most likely one of the most liked and respected series of all time. It is too hard to explain the vastness of the galaxy, but Douglas Adams can do it.

The English humor in this science fiction comedy book, is laugh-out-loud funny. That is, if you get it, and some do not. The series is inexplicably genious.

THe series is about the unfortunate events of planet earth and the events afterward, centering around two British characters: Arthur Dent and Ford Prefect. It begins on a Thursday... and with a house.

Basically, aliens in big, yellow spaceships come and destroy earth. Arthur Dent and Ford Prefect escape just in time to avoid being decimated with it. Their escapades through Magrathea, Earth, Krikket, and the Resturaunt at the End of the Universe can be very serious at times, but come out being awfully humorous. No one will ever replace Douglas Adams.

Reccomendation: ANYONE. The book is worth the money Amazon.com is asking for. So, all in all, I give it a 5 out of 5. People who do not like science fiction will love this book, along with the people who love science fiction. Douglas Adam's humor is universal. I know all who buy the book will enjoy it to no end.

Help other customers find the most helpful reviews

Was this review helpful to you? Yes No

Report abuse | Permalink

(a) Example review from Amazon.com (`http://www.amazon.com/review/R3HA8K97VUEYEW/`)


“Nice, comfortable hotel”

Reviewed December 6, 2009

1 person found this review helpful

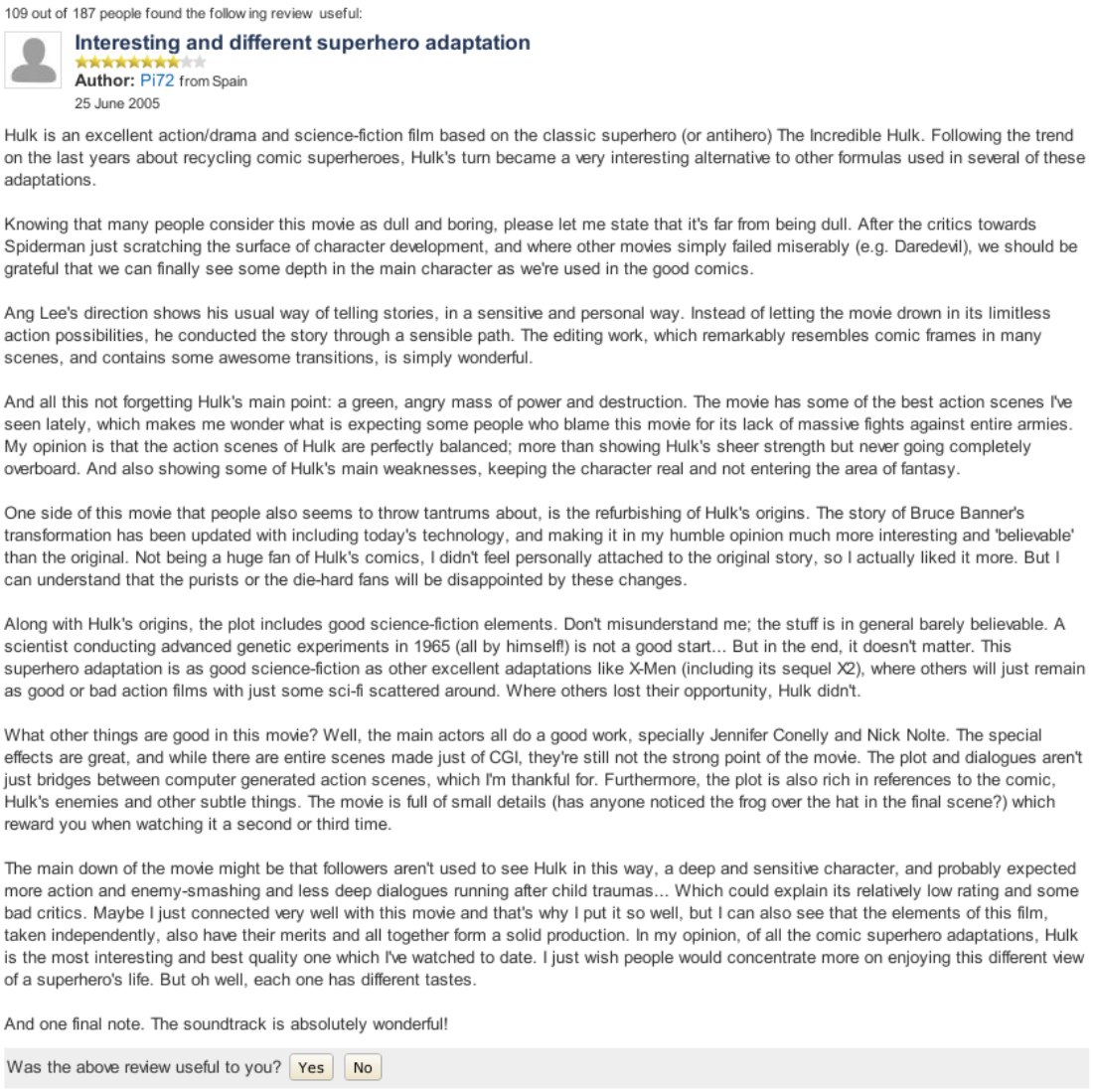
The hotel is in a residential neighborhood - approximately 15 minute walk to downtown area. We followed advice and took a taxi there from the train station, which cost about 7 euros - well worth it. The rooms are nice and bathrooms are modern. Good value for the money.

Stayed December 2009, traveled as a couple

Value
Location
Rooms
Cleanliness
Service

(b) Example review from tripadvisor.com (`http://www.tripadvisor.com/ShowUserReviews-g187291-d232577-r50749804-`)

Figure 3.1: Example reviews from different web services



109 out of 187 people found the following review useful:

Interesting and different superhero adaptation

Author: Pi72 from Spain

25 June 2005

Hulk is an excellent action/drama and science-fiction film based on the classic superhero (or antihero) The Incredible Hulk. Following the trend on the last years about recycling comic superheroes, Hulk's turn became a very interesting alternative to other formulas used in several of these adaptations.

Knowing that many people consider this movie as dull and boring, please let me state that it's far from being dull. After the critics towards Spiderman just scratching the surface of character development, and where other movies simply failed miserably (e.g. Daredevil), we should be grateful that we can finally see some depth in the main character as we're used in the good comics.

Ang Lee's direction shows his usual way of telling stories, in a sensitive and personal way. Instead of letting the movie drown in its limitless action possibilities, he conducted the story through a sensible path. The editing work, which remarkably resembles comic frames in many scenes, and contains some awesome transitions, is simply wonderful.

And all this not forgetting Hulk's main point: a green, angry mass of power and destruction. The movie has some of the best action scenes I've seen lately, which makes me wonder what is expecting some people who blame this movie for its lack of massive fights against entire armies. My opinion is that the action scenes of Hulk are perfectly balanced; more than showing Hulk's sheer strength but never going completely overboard. And also showing some of Hulk's main weaknesses, keeping the character real and not entering the area of fantasy.

One side of this movie that people also seems to throw tantrums about, is the refurbishing of Hulk's origins. The story of Bruce Banner's transformation has been updated with including today's technology, and making it in my humble opinion much more interesting and 'believable' than the original. Not being a huge fan of Hulk's comics, I didn't feel personally attached to the original story, so I actually liked it more. But I can understand that the purists or the die-hard fans will be disappointed by these changes.

Along with Hulk's origins, the plot includes good science-fiction elements. Don't misunderstand me; the stuff is in general barely believable. A scientist conducting advanced genetic experiments in 1965 (all by himself!) is not a good start... But in the end, it doesn't matter. This superhero adaptation is as good science-fiction as other excellent adaptations like X-Men (including its sequel X2), where others will just remain as good or bad action films with just some sci-fi scattered around. Where others lost their opportunity, Hulk didn't.

What other things are good in this movie? Well, the main actors all do a good work, specially Jennifer Conelly and Nick Nolte. The special effects are great, and while there are entire scenes made just of CGI, they're still not the strong point of the movie. The plot and dialogues aren't just bridges between computer generated action scenes, which I'm thankful for. Furthermore, the plot is also rich in references to the comic, Hulk's enemies and other subtle things. The movie is full of small details (has anyone noticed the frog over the hat in the final scene?) which reward you when watching it a second or third time.

The main down of the movie might be that followers aren't used to see Hulk in this way, a deep and sensitive character, and probably expected more action and enemy-smashing and less deep dialogues running after child traumas... Which could explain its relatively low rating and some bad critics. Maybe I just connected very well with this movie and that's why I put it so well, but I can also see that the elements of this film, taken independently, also have their merits and all together form a solid production. In my opinion, of all the comic superhero adaptations, Hulk is the most interesting and best quality one which I've watched to date. I just wish people would concentrate more on enjoying this different view of a superhero's life. But oh well, each one has different tastes.

And one final note. The soundtrack is absolutely wonderful!

Was the above review useful to you? Yes No

(c) Example review from imdb.com
(`http://www.imdb.com/title/tt0286716/reviews`)

be less rich in clues. For example, consider this (complete) document:[6]

(3.5) *This thermometer lasted about a month before going in to permanent "Hi" mode. What a ripoff!*

The only direct clue in this document is *ripoff* which, in a supervised approach, must occur in the training data in order to correctly classify the example.

Supervision through User-Supplied Ratings A strategy frequently employed in document-level sentiment analysis is to use the user-supplied ratings for supervision. Often, the ratings are first mapped to a common simplified scale (e.g., 5 stars to binary). Using the user's rating as the label of a review is an appealing idea as it makes annotated data available in vast amounts and thus eliminates any need for manual annotation. It has been argued, however, that using product ratings as labels for the accompanying texts constitutes a form of noisy labeling. Ganu et al. (2009), for example, show that star ratings are far from being perfectly correlated with sentiment annotated by readers, and that manually annotated reviews produce better recommendations.

There are many possible causes to this effect. First, there are usually no standards to which users have to adhere. Some online reviewing systems associate star ratings with verbal assessments (e.g., *very good*, *very bad*, *great* or *disappointing*), but usually, there are no detailed guidelines that give specific instructions for the rating process. Second, some readers see the review text as complementary to the rating, and thus, the rating may be inconsistent with the views expressed in the text. These results and observations show that although sentiment classification is often described as supervised, caution must be taken with viewing star ratings as a gold standard.

Overall, the document-level approach has been found to be too coarse for many practical applications. Thus, finer-grained analysis levels have been proposed, which we review next.

[6] Example taken from the Multi-Domain dataset, kitchen domain.

3.2.2 Sentence/Clause Level

Documents often exhibit a clear overall polarity (likely to be expressed in the concluding portion, cf. Pang and Lee, 2004). Smaller units such as sentences or clauses are significantly harder to analyze. The task of predicting the polarity of a sentence is again a problem of text classification, where the text, represented through word-based features, is the input to a statistical model. Currently, the most successful clue-based approach uses a combination of Naive Bayes with a Support Vector Machine (Wang and Manning, 2012). As described previously, in this setup, sentiment analysis relies on the availability of clue words. However, due to the significantly shorter length of sentences, not every sentence may actually contain a useful clue. Consider the negative sentence[7]

(3.6) *Might best be enjoyed as a daytime soaper.*

where the reader has to understand that the movie in question is similar to a daytime soap, which is a somewhat derogatory term for TV show of low production quality for casual entertainment. In addition, the sentence contains the word *best*, which has potential to be mistaken as positive in the clue model. Many sentences also require the resolution of hedges and conditionals, such as

(3.7) *Fans of Plympton's shorts may marginally enjoy the film.*

(3.8) *This book fills a much-needed gap.*[8]

Sentences may also be ambiguous when read out of context, or may contain expressions of mixed sentiment. An early formal model for coping with complex polarity structure was introduced by Polanyi and Zaenen (2006). In their model, polarity-bearing clues are modified through so-called *polarity shifters*, being intensified, diminished, or reversed. For example, *marginally* in sentence 3.7 diminishes the polarity of *enjoy*. Note that while the shifter model works well for sentence 3.7, sentence 3.8 is more difficult as *gap* is an uncommon clue, and the shifters *much-needed* and *fills* are highly specific to that clue (i.e., they would not be shifters in other contexts).

[7]Example taken from the Pang & Lee sentence dataset.
[8]A quote attributed to Moses Hadas.

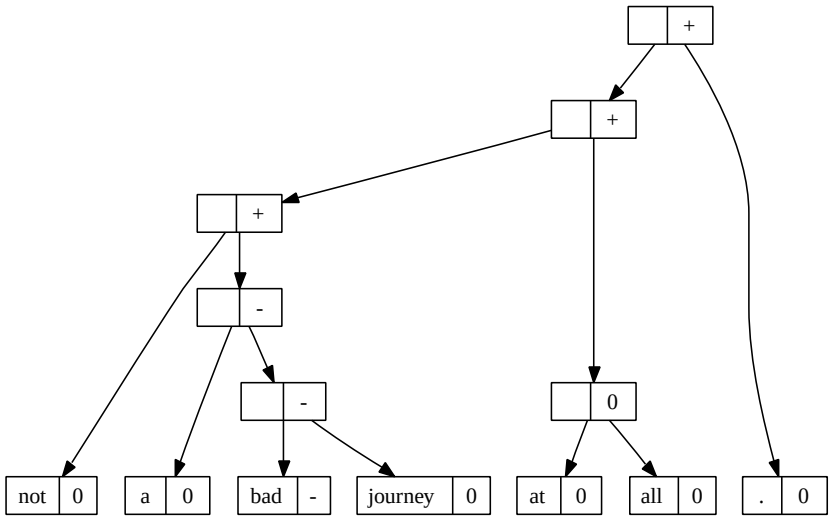


Figure 3.1: Compositional analysis of the sentence “Not a bad journey at all.”. Each node in the tree has positive (+), neutral (0), or negative (-) polarity. The polarity of leaf nodes may be lexical while higher-order nodes receive they polarity by composition of their children. Thus, “bad” has negative polarity, and the phrase “not a bad journey” has positive polarity.

To generalize this concept, it has been suggested that sentence sentiment follows a hierarchical *compositional* structure, formalized for example through constituency (Fahrni and Klenner, 2008) or dependency trees (Nakagawa et al., 2010). In the resulting tree, each node is assigned a polarity; the overall sentence polarity can be read at the root node. Non-terminal polarities are the result of shifter-like operations recursively carried out at each node. An example analysis in the constituency framework can be found in Figure 3.1. Such structures may be generated with a manually defined grammar (e.g., Fahrni and Klenner, 2008) or learned automatically, either unsupervisedly (e.g., Socher et al., 2011) or from annotated example trees (Socher et al., 2013).

3.2.3 Entity/Aspect Level

Liu defines an *entity* as any object about which sentiment is expressed. An entity thus constitutes a type of *target* which was introduced previously. An *aspect* is any property or part of the target, for example the lens of a camera. An example for aspects can be found in Figure 3.1b where the user rated various aspects of his stay at the hotel, such as location and service. We discuss the notions of entities and aspects only briefly as they are beyond the scope of this thesis.

Sentiment analysis of entities or aspects does not focus on a single linguistic unit for analysis. Instead, all information about the entity is collected and used for making a prediction. This task has been formalized as *fine-grained sentiment analysis* (e.g., Yang and Cardie, 2013) where the relation between opinions, targets, and sometimes opinion holders have to be recognized. In contrast to the problems discussed above, this task involves structured prediction comparable to other NLP problems such as semantic role labeling.

Liu argues that analysis on the entity level is superior to considering individual units (such as single phrases or sentences) as knowledge about the target is necessary for resolving ambiguities. Conversely, entity-level sentiment analysis requires holder and target detection which leads to a significantly more complicated machine learning task.

3.3 Feature Representations

Word-based clue representations are the most important features in sentiment analysis. Despite the fact that sentiment exhibits complicated syntactic properties, bag-of-word representations are sufficient for high-accuracy classification: For in-domain classification of binary polarity, accuracies of up to approximately 90% for documents and between 80 and 85% for sentences have been achieved using only word features (Wang and Manning, 2012). Note that the results may vary depending on the domain and the environment (e.g., news vs. web data).

Research are also appealing due to the fact that low-level linguistic entities (e.g., words or phrases) can express sentiment and may thus be classified the same way as higher-level entities (e.g., documents). This enables joint models of sentiment on different levels. For example, when using a bag-of-words features, we can exploit this fact by labeling the features with sentiment directly (e.g., through a polarity lexicon). This feature knowledge could then be incorporated into a statistical model, for example as a prior in a Bayesian model (e.g, Melville et al., 2009; Settles, 2011). Using clues to label larger example types constitutes a form of distant supervision where the polarity lexicon is the supervision database.

While some investigations into linguistic features for document classification have been conducted (Gamon, 2004; Ng et al., 2006), the common result so far has been that most intuitively helpful linguistic features such as syntactic relations do not increase the quality of sentiment analysis in practice. In contrast, syntactic kernels have yielded promising improvements for sentence classification (Matsumoto et al., 2005). The intuition behind such models is that shifters can be modeled with the syntactic structure. However, such approaches rely on clean syntactic analysis which is not easy on web data (Foster et al., 2011).

3.4 Summary

In this chapter, we gave an introduction to sentiment analysis. We first described concepts that relate to sentiment. Next, we introduced automatic sentiment analysis on the document, sentence, and entity levels. Finally, we discussed feature representations for sentiment analysis.

4 Sentiment Classification with Active Learning

4.1 Introduction[9]

The predominant approach to document-level sentiment classification is supervised learning, which depends on annotated training examples. Unfortunately, annotating data is an expensive and time-consuming process which traditionally involves paid labor of *domain experts* – humans who (i) understand the task at hand, (ii) speak the language in which the data is written, (iii) are capable of using an appropriate user interface for annotation, and (iv) work according to some predefined standards. In sentiment analysis, it is possible to exploit user-supplied ratings for supervision. It has been argued, however, that is not always an advisable strategy (Ganu et al., 2009), as criterion (iv) is usually violated in online reviewing systems – there are simply no guidelines. Unfortunately, guidelines play an important tool for achieving high annotation quality (Pustejovsky and Stubbs, 2012). Thus, annotating polarity for sentiment analysis in a more rigorous environment would be generally desirable.

The way training data is selected has a strong influence on the performance of a supervised system. Supervised, clue-based sentiment classifiers are often difficult to improve after a certain point as the clues occurring in documents are highly redundant. However, there are also many documents which contain only unseen clues – for example very short documents – on which the classifier fails to make correct predictions. To address this problem, it would be helpful to train the model on a training set that offers high coverage of the test data. This motivates the use of active learning, a technique where the classifier has a direct influence on the process of selecting data for annotation.

As data annotation is expensive, we are interested in a cost-efficient solution.

[9]This chapter describes joint work on active learning with Florian Laws. The author of this thesis carried out the sentiment analysis experiments.

There are two parameters that influence the overall cost of a trained system. The first is the *individual cost* of a training example, i.e., the amount of money we have to spend to get one example annotated. The second is the *number of examples* we need to annotate. Lowering any of these amounts will result in a lower overall cost. Of course, we want to lower overall cost without negatively affecting the system's accuracy.

To lower the individual cost, we can resort to non-expert annotators. Recently, various web services that serve as marketplaces for tasks have emerged. One such service is Amazon Mechanical Turk (MTurk), a web platform for offering small tasks on the web to workers. MTurk workers are usually willing to annotate at very competitive price levels, going as low as $0.01 for an example. Regarding the criteria for experts stated above, we find that on MTurk, (i) is less critical for sentiment analysis of documents as humans are natural experts at understanding sentiment, thus assessing document-level sentiment is a straightforward task. (ii) may be a problem depending on the language in question as MTurk users are mostly English-speaking. We will thus conduct experiments on English data. Note, however, that we have little control over the actual skills of MTurk annotators. (iii) depends on the task. We will focus on document annotation, where a simple binary decision has to be made. Other tasks, particularly those involving relations of entities (e.g., fine-grained sentiment analysis) may not be as easy to accomplish on MTurk. (iv) may again be problematic for complex tasks, as long annotation guidelines may potentially be ignored by workers.

Lowering the number of examples is not as straightforward. Simply using fewer training examples will most likely lead to lower system accuracy. As mentioned above, to get good coverage of clues, it would be an advantage to pick the most informative examples from our data and eliminate all redundancies. In general, however, finding a subset of the data that fits this criterion is difficult. Often, researchers apply heuristics to select a minimal subset of interesting examples, however this selection method could over-represent some phenomena and under-represent others that the researcher missed. This is particularly problematic in the clue model as it is hard for humans to guess which clues would be helpful to the system. To eliminate direct human influence on the selection process, it has been proposed to automate the selection process. We could let the system select

informative examples for us by using some measure of confidence. We would then get these examples annotated by humans and retrain the system on the new training set, after which new unlabeled examples could be re-rated by the classifier. This process is an instance of *active learning (AL)*. As the system suggests informative examples, we would expect a better performance level on fewer annotated examples than if we selected examples randomly.

While both active learning and crowdsourcing are popular techniques in NLP, there is little research on whether the two work in combination. Combining them, however, is promising as we would expect large cost reduction since the two techniques are orthogonal. Most crucially, annotators on MTurk are not domain experts in general. Therefore, they produce noisy labels which can be a problem for active learning. We believe, however, that they should be able to perform a review annotation task with little difficulty. Nevertheless, we can never be sure that all workers carry out their work responsibly, particularly with spam being a frequently reported issue on MTurk. For this reason, we will evaluate annotation quality and investigate ways to improve it.

We have another interest in document-level sentiment analysis besides obtaining high-quality sentiment labels. Statistical sentiment classifiers using clue features essentially learn the importance of each clue for a class during training. Active learning influences the way these clues are learned since the classifier should pick examples that help to disambiguate clues. We would like to analyze this behavior and how the size and sample of this data influence it.

The rest of the chapter is structured as follows. First, we introduce the methods used in our experiments in Section 4.2. In Section 4.3, we describe our experiments and discuss the results. Finally, we give an overview of related work in Section 4.4.

4.2 Methods

4.2.1 Crowdsourcing with Amazon Mechanical Turk

Amazon Mechanical Turk (MTurk)[10] is a web service for online work. *Requesters* can post bulk tasks which are then offered to *workers*. It is required that the

[10] `http://www.mturk.com/`

tasks can be broken down into small, repetitive instances, called *HITs* (Human Intelligence Tasks). For example, part-of-speech annotation of a corpus can be broken down into individual sentences, each of which constitutes a HIT. These are then offered to workers, where the number of workers a_i that each work on the same HIT x_i is a parameter the requester has to specify. Workers can decide for each HIT whether or not they want to work on it. This process is often referred to as crowdsourcing as the task at hand is outsourced to a crowd.

In practice, the set of workers may change quickly, with some workers annotating only a few hits and others annotating large quantities. We will investigate this in more detail in our experiments. Of course, this fluctuation may mean that we will get many annotations by completely untrained workers who have no experience on our task. For this reason, we will establish a way of quality control.

There are two different models of how requesters interface with MTurk. In *batch* mode, the requester designs an HTML template with a number of variables which are later instantiated by the individual HITs that the requester uploads to MTurk as a table. This setup is straightforward to use, however it is also relatively static as all aforementioned parameters have to be set before the experiment even begins. Particularly, all a_i must be equal. The template, and thus the number of variables, is fixed and cannot be customized for each example. Further, we have no control over the order in which MTurk offers the HITs to the workers. These points are problematic for an active learning experiment as we want to avoid selecting data beforehand.

External questions are the alternative to this setup. Here, the requester sets up an HTTP server which is then registered with MTurk. After posting HITs, the server returns a HTML page for a HIT once it is requested. MTurk includes both a HIT ID and a worker ID in the request. The HTML page for each hit can thus be created dynamically during the course of the experiment. In particular, the requester does not have to specify a list of examples beforehand and instead has control over which worker gets which example. This setup is much more flexible but also involves significantly more work for the requester.

There are several parameters that determine the cost of an experiment on MTurk:

- The **number of HITs** n we post in total, i.e., the number of examples we get annotated.
- The **number of workers per HIT** a_i we require to work on a HIT x_i, i.e., the number of annotations we get for each example.
- The **reward** r **paid to each worker for a HIT**.

The amount we pay for an experiment is the *budget*

$$B = \sum_{i=1}^{n} a_i \cdot r \cdot m, \tag{4.1}$$

which can be simplified to $B = n \cdot a \cdot r \cdot m$ in the batch case. The parameter $m = 1 + c$ introduces $c = 0.1$, the commission paid to Amazon, into the equation. For example, if we have $n = 10$ HITs for which we would like $a = 3$ workers each at $r = \$0.01$ per HIT, we would spend a total of $B = 10 \cdot 3 \cdot \$0.01 \cdot 1.1 = \$0.33$ of which \$0.30 are paid to workers and \$0.03 to Amazon.

Quality control is important when annotating data, and even more so if we outsource this task to anonymous workers. MTurk offers basic capabilities for quality control. Before the experiment begins, a requester may put in place several restrictions for his experiment. Each worker has an *approval rate*, which is the ratio of HITs he got accepted (see below). Requesters can set a minimum approval rate threshold on their hits to exclude bad workers. Additionally, it may be required that the workers be from a specific country (e.g., only workers from the USA), or that the workers have a set of qualifications which can be obtained by passing tests set up by requesters. Naturally, these restrictions will reduce the number of workers who approach a task. After the experiment, requesters can reject submissions with inferior quality, influencing the workers' approval rating. However, this option must be used carefully as wrongfully rejecting many workers may result in bad reputation for the requester. This system seems reliable at first, however due to the anonymity of workers, it may be exploited by creating multiple accounts or pushing one's agreement rate by working on self-submitted HITs.[11] Recently, MTurk has introduced *Masters*, a subset of workers that Amazon considers highly

[11] `http://www.behind-the-enemy-lines.com/2010/10/be-top-mechanical-turk-worker-you-need.html`

qualified according to their accuracy on certain image and text processing tasks. It is however controversial whether Masters perform better than the general set of workers (Kandasamy et al., 2012; Difallah et al., 2013). At the time of our experiments, Masters had not been introduced.

It has also been noted that spam is an increasing problem on Mechanical Turk (Ipeirotis et al., 2010). Some spammers even use automated means or work in groups to circumvent countermeasures by requesters. This emphasizes the need of good quality-control measures on MTurk.

The optimal reward a requester should for a HIT is a difficult parameter to determine on MTurk. If the reward is too low, workers might be unwilling to participate. If it is too high, the cost advantage of using MTurk might disappear. Fort et al. (2011) note that in several experiments, rewards were not correlated with annotation quality. While better compensation may motivate workers to perform better, HITs with high rewards also tend to attract spammers.

4.2.2 Annotation System

An important requirement for an active learning implementation is that it enables fast annotation-re-training cycles to minimize wait times for annotators and staleness. We employ the annotation system by Laws (2013) which is capable of performing annotation and classification concurrently. The system is strictly separated into a front end and a back end.

The system's front end interfaces with MTurk workers through external questions. If a worker requests a HIT, the system generates an HTML page based on which annotation is required next and serves it through HTTP. The back end of the system manages annotations and controls the classifier process. It offers a choice of different AL strategies, including uncertainty selection from a pool which we use for our experiments.

In the pool-based strategy, the back end keeps the unlabeled data in a queue which is sorted decreasingly by the uncertainty of the examples. If an example is needed for annotation, it is taken from the head of the queue and passed to the front end for display. Once an example has been annotated, it is removed from the queue and added to the training set. The classifier is re-trained as often as possible

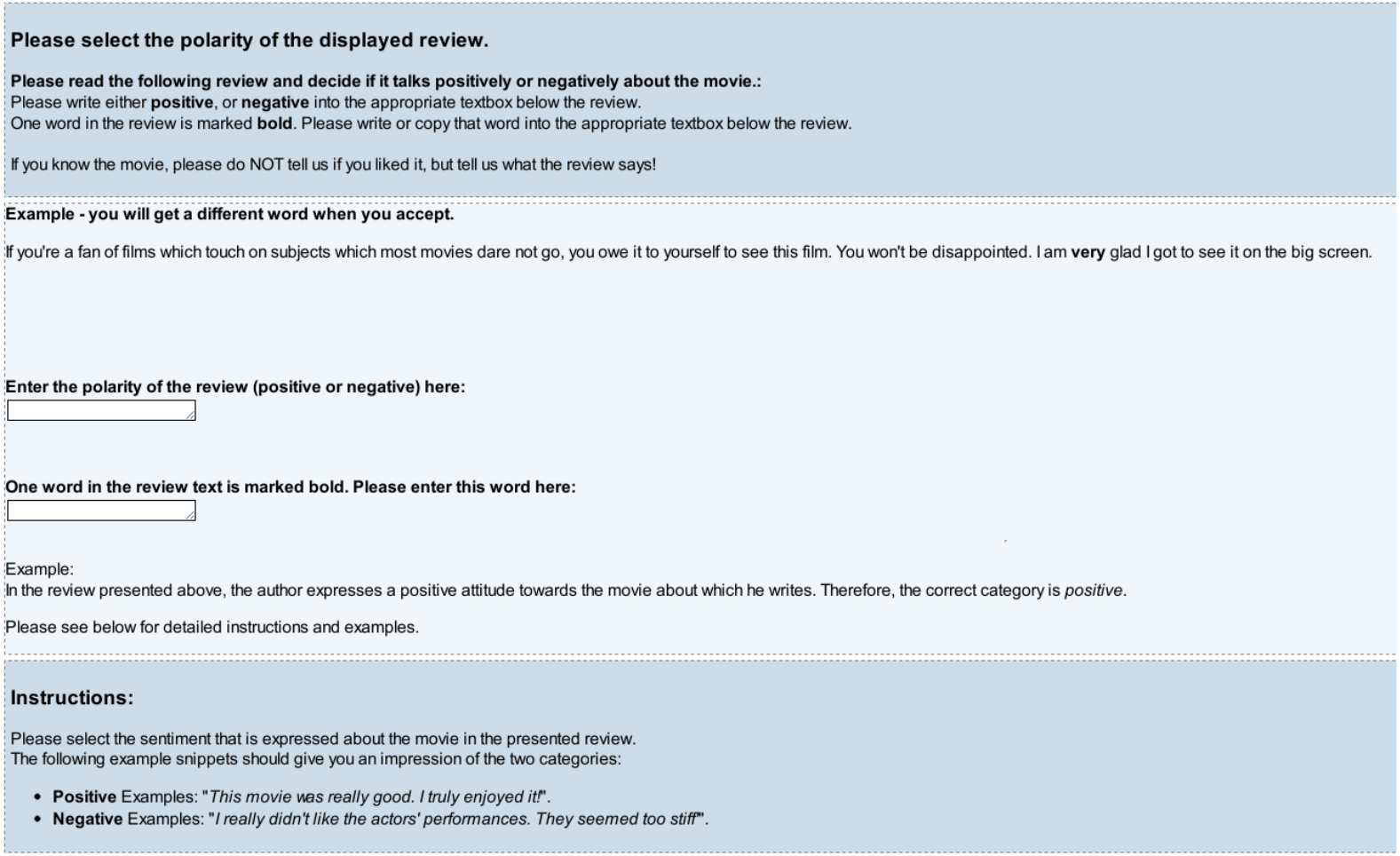


Figure 4.1: Example HIT as displayed on MTurk. Workers have to type out the sentiment label as well as a randomly selected word from the review. The random word is marked bold in the review text.

and necessary. Generally, there is no guarantee that the system is done with retraining before the next example arrives. The queue can be reordered concurrently by applying the classifier to the data and re-calculating uncertainties. After the reordering of the examples in the queue, we have selected a new batch of examples. The queue strategy can thus be viewed as a batch strategy where we intend to keep batches as small as possible.

4.2.3 User Interface

We created a user interface for sentiment annotation, shown in Figure 4.1. The top part of the form contains a brief task description, followed by the example selected for annotation, and the input area. We present one document per HIT. Underneath the input area are more detailed annotation guidelines. They contain a short description for each label, along with examples. For our initial experiment, we implemented the label selection with radio-buttons, which proved to be a weak point that could easily be exploited by spammers. We therefore introduced a modified user interface which we introduce next.

4.2.4 Quality Control

Bad workers and spammers are a massive issue on MTurk. There are many individuals who try to game the system by submitting bad HITs, while others simply do not understand the task or the language well enough. We are therefore interested in taking measures to ensure high annotation quality.

Adaptive Voting

On MTurk, we are dealing with noisy labels from untrained annotators. In such situations, it may be harmful to rely on the decision of a single annotator. In cases where each annotator assigns a single label to an example, it is natural to take a committee approach. This means that we would obtain multiple competing labels for each example based on which the final label is selected through *voting* (each annotator votes for a label). This method has two parameters: the number of annotations we obtain, and the type of majority we require to accept a label

(e.g., more than 50% of annotators choosing the label).

The system by Laws (2013) offers a generalization of this approach, called *adaptive voting*. Here, we iteratively request labels from different annotators until acceptance or rejection criteria are reached. These criteria depend on the following parameters:

- α, the threshold of agreement
- m, the minimum number of votes
- d, the maximum number of votes

We obtain $m \leq a_i \leq d$ labels for each example x_i, referring to this process as d-voting. If the ratio of annotators voting for the most frequent label is greater than or equal to α once $a_i \geq m$, we accept the label. We discard the example if we have not reached agreement after d annotations.

When using adaptive voting, we have to wait until we reach agreement between the annotators. As proposed by Laws (2013), we re-train the classifier using *tentative* labels with majority voting even before the voting criteria are fulfilled. When the voted annotation is available, the tentative label is replaced.

Counter-Spam Measures

We carried out preliminary experiments (Section 4.3.2) and found that an interface with radio buttons for the document label (`positive` or `negative`) leads to a high number of spam submissions which hurts the classification results. In this setup, we achieve a classification accuracy of only 55%, which is slightly above chance.

We re-designed our template to make automated submissions more difficult. Annotators have to type out the label, and, in addition, a randomly selected word from the document, as shown in Figure 4.1.

Additionally, we observed that spammers would often resort to submitting the same answer over and over again to all HITs. As it is very unlikely that the system actually selects examples with the same label repeatedly, we could perform a binomial test to check whether the set of labels submitted by the worker deviates from a uniform distribution. We first collect a sufficiently large number (k) of examples

from an annotator. We then calculate the relative frequency $r = c_1 / \sum_i c_i$ of the most frequent label c_1 submitted by the worker. If, given a threshold t_r, $r > t_r$ over the first k HITs submitted by the worker, the worker is a suspected spammer and will be blocked from working on further HITs. Assuming the true label distribution is indeed uniform, this algorithm will make a false positive decision with a probability of $P_{\text{Bin}}(X \geq kt_r)$.[12]

With these two simple techniques in place, we were able to improve the average individual worker-accuracy to around 75%, and the average voted label correctness to around 90%.

4.3 Experiments

4.3.1 Experimental Setup

We chose a well-known document analysis setup for the sentiment classification task of (Pang et al., 2002) on the movie corpus (described in more detail in Appendix A.2.1). This dataset contains 1000 positive and 1000 negative movie reviews. The task is to predict binary document-level polarity.

We split this corpus randomly into an active learning pool consisting of 1500 reviews and a test set containing 500 reviews. We use a maximum entropy (MaxEnt) model for classification (Stanford Classifier implementation, cf. Appendix A.1.1). We extract bag-of-words feature representations from the documents. As the dataset is balanced, we evaluate classifier accuracy. We check statistical significance of accuracy improvements with the randomized approximation test (Section 2.6.2), using a significance level of $p < 0.05$. To find out whether active learning is effective, we compare it to *random sampling* where we just select documents randomly. We use the same parameter settings for both experiments.

We set up experiments as follows. At the beginning, the system starts with a set of two randomly sampled documents, one for each class, which are annotated with gold labels. We put the remaining 1498 documents in the unlabeled pool. These documents are then iteratively selected for annotation through uncertainty selection and random sampling, respectively, and posted as HITs on MTurk. Each

[12] Using our setup of $t_r = 0.9$ and $k = 20$, we implicitly conduct a binomial test at $p \leq 0.001$.

HIT contains a single document, and we pay a reward of \$0.01 per hit. For adaptive voting we use an agreement threshold of $\alpha = .75$ and voting parameters $m = 2$ and $d = 4$ (4-voting). In addition, we simulate the case where each example receives only a single annotation by selecting the first annotation for each example. This is to reduce the number of experiments we run on MTurk. Of course, the results for this experiment will not be entirely accurate as selection was performed by a classifier using voted labels. The threshold for automatically blocking spammers from experiments is set to $t_r = 0.9$ which is calculated after $k = 20$ examples. This means that workers who submit the same label more than 90% of the time are considered spammers. We also require all workers to have an approval rating of 100%.

In total, we conducted two runs of each setup (uncertainty selection and random sampling). The respective runs were started two days apart from each other in order to avoid having too many workers participate in both experiments. We always compare runs at equal *budgets*, i.e., the total reward we pay. As we always pay the same reward for all HITs, we simplify the calculation of the budget to

$$B = \sum_{i=1}^{n} a_i, \tag{4.2}$$

i.e., the sum of the counts of annotations a_i that we obtained for each example x_i.

4.3.2 Results

Annotation Quality

Annotation quality is a critical point in NLP experiments – even more so when using crowdsourcing, since annotation is carried out by non-experts. To evaluate annotation quality, we first define measures for worker performance. Since we have gold annotations available for our data, we can compare them to the workers' annotations directly. We define *label accuracy* (LA) as the relative frequency of correctly annotated documents according to our gold standard. *Worker accuracy* (WA) is the label accuracy for the subset of documents annotated by a certain worker. Finally, we also look at *classification accuracy* (Acc), measuring the amount of correctly classified documents we achieve with a classifier trained on this data.

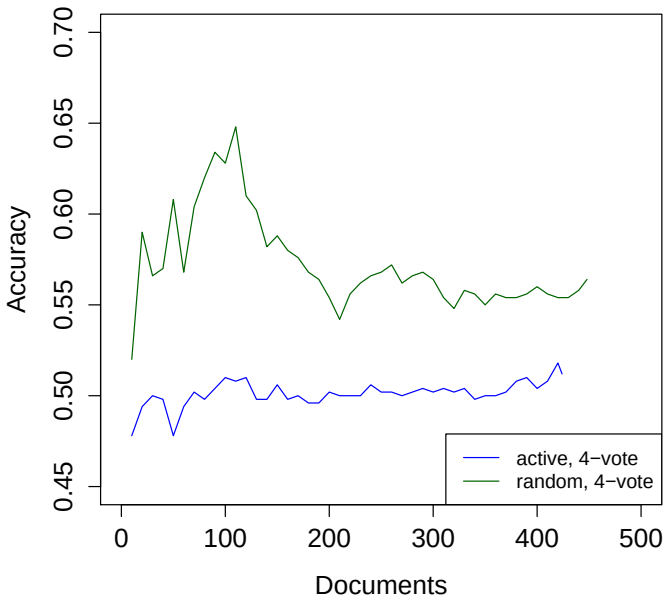


Figure 4.2: Active learning vs. Random sampling without counter-spam measures

Classification accuracy of course depends directly on the quality of the annotated training material.

Note however, related to the argument from Section 3.2, that star ratings might not be perfect gold annotations. Thus, reviews annotated by humans may actually have higher quality than taking the star ratings as gold data which is the case with this dataset. With this argument in mind, we would not expect perfect agreement between annotated and gold labels.

Without counter-spam measures We first run an experiment without using our counter-spam measures in place. Figure 4.2 shows the classification results we achieve with this setup. Generally, classification accuracy is low, around 55% for random selection and 50% for active learning.

This problem is caused by spam submissions. We encountered two types of spam-

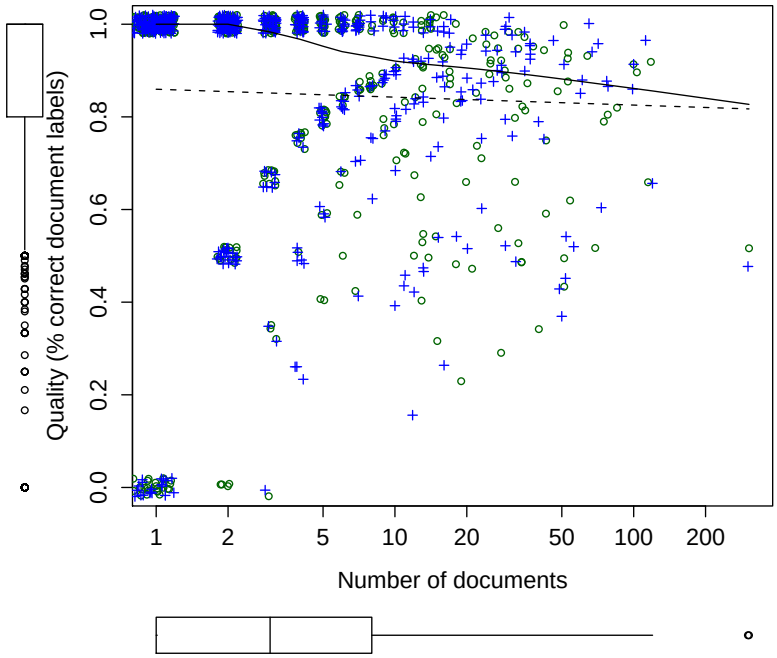


Figure 4.3: Worker accuracies of all workers from all runs. Each point represents a single worker. Blue crosses: run 1, green points: run 2. Solid line: linear regression, dashed line: locally weighted regression.

mers. One selects classes randomly, the other always selects the same class. Most notably, one worker submitted around 600 positive labels (and only 50 negative), around half of which were incorrect. One reason for why the active learning classifier performs even worse than random selection is that the selection process is misled by wrongly labeled examples. Since active learning selects examples with high uncertainty, a wrong label will have more impact on the classifier and thus influence the uncertainty scores of all examples in the pool, causing further errors.

Spam also leads to high disagreement between the annotators. We had to discard around 43% of the documents because they are below the agreement threshold.

With counter-spam measures After enabling counter-spam measures, we check annotation quality at a budget of 1130. Figure 4.3 shows worker accuracy jointly for all workers from all runs. We plot worker accuracy in relation to the number of examples the individual worker has annotated in total. We find that, in general, there are many workers who submit only correct labels. However, we also note that accuracy decreases somewhat for workers who submit many HITs. It is difficult to determine whether those workers are simply getting tired after annotating many documents or whether they are intentionally submitting spam. We believe, however, that the two points at the far right of the plot are spammers as they submitted a high quantity of HITs, achieving only around 50% accuracy, which is the expected performance when guessing.

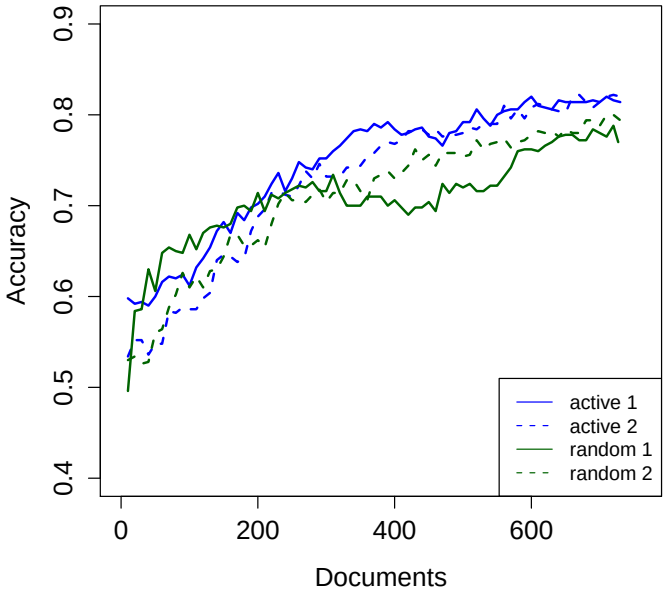
Label accuracy for single annotations (shown in Table 4.1) is between 74.8% and 76% at this point on average for the RS and AL runs, respectively. This shows that it is indeed difficult to obtain high-quality annotations from single workers on MTurk. We will later show that this level of label accuracy actually leads to inferior classification accuracy. After applying 4-voting, we get an increase to 89% label accuracy for both settings.

The relative frequency of discarded examples is much lower in this experiment, around 28%. This means that there is much more agreement between the annotators.

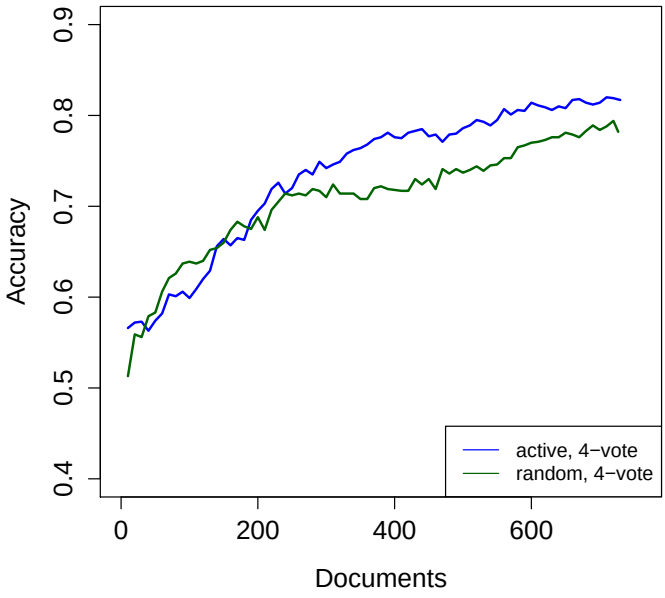
Random Sampling vs. Active Learning

For our first experiment on classification quality, we compare the results from the active learning (AL) and random sampling (RS) setups. We compare random sampling and active learning at equal budgets. We report numbers for three budgets, 1130, 1500 and 1756, in Table 4.1.

For the smaller budget of 1130 examples, we see a significantly improved performance when using active (line 4) learning instead of random sampling (line 2). In fact even the single voting active learning classifier (line 3) is significantly better than the random sampling classifier using adaptive voting (line 2) by 3.6%. While we still see a significant advantage of active learning over random sampling at a budget of 1500, it is not significant anymore at 1756.



(a) 2 individual runs for active learning and random sampling



(b) Means over both respective runs

Figure 4.4: Active learning vs. Random sampling

Budget			1130				1500		1756	
			#train	Acc	cost/doc	LA	#train	Acc	#train	Acc
RS	1	S	1130	70.4	1	74.8	–	–	–	–
	2	4-v	450	71.2	2.51	89.6	600	76.0	735	79.2
AL	3	S	1130	$74.8^{\dagger\ddagger}$	1	76.0	1500	73.2	–	–
	4	4-v	455	$77.4^{\dagger}$	2.48	89.0	604	$81.0^{\dagger\ddagger}$	715	81.8

Table 4.1: Results for sentiment classification on the movie corpus with different experimental setups. #train = size of training set (documents), LA = label accuracy. S = single, 4-v = 4-voting. Budgets 1130 and 1500 shown for run 1; budget 1756 is an average over 2 runs. † indicates that the F_1 result is a statistically significant improvement over the corresponding RS result, ‡ over the previous line (approximate randomization test, $p < 0.05$).

Figure 4.4 shows the overall performance of active learning compared with a baseline of randomly selected documents. Active learning consistently performs better than random sampling after annotating around 300 documents. The active learning run 2 performs worse than run 1 because the annotation quality is lower at the beginning of the experiment. We find that these results exhibit similar behavior to simulated active learning (cf. Li et al., 2012). The curve rises steeply at the beginning and flattens as more labeled examples are acquired. At around 600 documents, the active learning curve reaches a somewhat steady level as random sampling catches up, and, as described above, the difference between the methods is not statistically significant anymore.

The initial gains through active learning are small. There are several possible reasons for this. First, we have a small set of labels compared to other tasks such as Named Entity Recognition. Second, there is a set of common clues that covers a high number of examples (cf. Wilson et al., 2005b). As they occur in many examples, it is possible to find a sufficiently large set quickly through random sampling. After some time, uncertainty selection will disprefer redundant examples and thus, examples containing unseen clues are selected instead. We will come back to this effect during error analysis.

We also measured the staleness of the examples in the queue. It turns out that

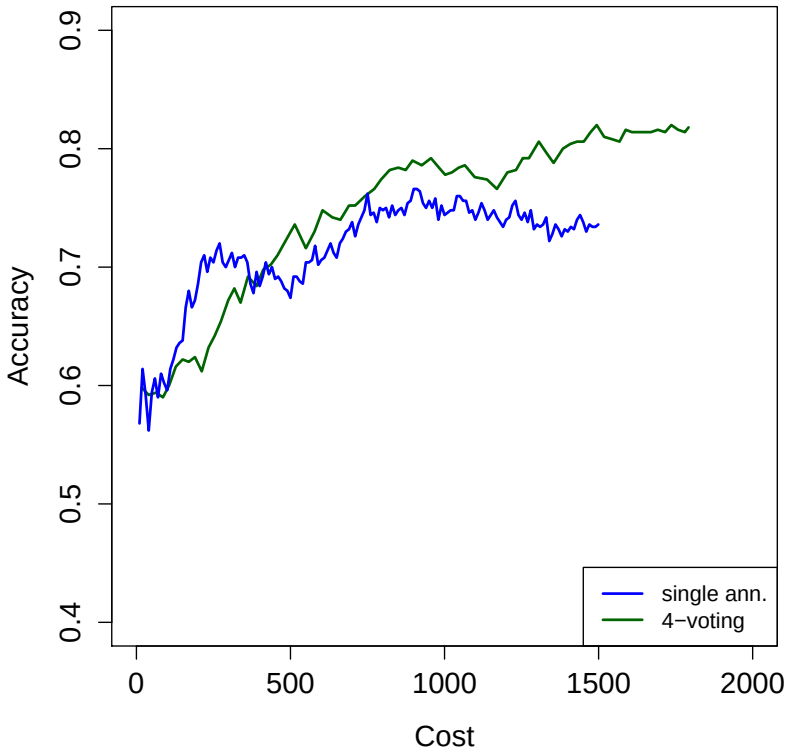


Figure 4.5: Adaptive voting vs. single annotation on AL run 1

re-training the sentiment classifier is fast enough so that the re-ordering of the queue finishes before the next example is requested most of the time. This means that we have an average batch size close to 1.

Adaptive Voting vs. Single Annotation

Using a voting strategy with a fixed budget means making a trade-off between the number of examples annotated and the number of annotations obtained per example. Increasing the size of the training set by having more examples annotated is useful because it would increase diversity in the training set. On the other hand, having more labels for each training example might increase annotation quality, making the training data less noisy, increasing the cost of each example. Thus,

finding the right trade-off between quantity and certainty is difficult.

We carry out experiments comparing two voting settings, one with 4-voting and one using a single annotation. We generate the single annotation data from the voted data of run 1. This means that we have the same total number of annotated documents for both experiments. However, the budget will be lower for the single annotation experiment when using the same amount of data, so we cannot compare the results for the full budget. Instead, we use two smaller budgets. The first one, 1130, taken from the previously collected data for both active learning and random sampling. We also extended the active learning run to 1500 single annotations, which we obtained from MTurk in a separate experiment. Results are shown in Table 4.1. At 1130, we find that there is an improvement of 2.6% when using 4-voting instead of single annotations for active learning, and 0.8% for random sampling. These improvements are not statistically significant. At a budget of 1500, we obtain a significant difference of 7.8%. Note that the single annotations still use 4-voting selection, which might have had a positive impact on the results.

Figure 4.5 shows accuracy over cost for the two annotation strategies. The results show that, initially, accuracy increases more rapidly for single annotations, possibly due to the fact that the classifier simply is trained on much more data (around 2.5 times as much, as the cost per annotation in Table 4.1 shows). However, the single annotation run plateaus much earlier than the 4-voting run. From Table 4.1 we can see that the voted label accuracy is significantly higher the one for single annotations. In the long run, a small set of high-quality annotations seems to produce a better classifier than a large set of noisy annotations. One possible cause of this effect could be that active learning finds fewer informative examples at late stages, so voting improves performance by getting the most informative examples right. Additionally, single annotation is of course more susceptible to spam submissions. We saw earlier that spam has a considerable impact of the active learning process.

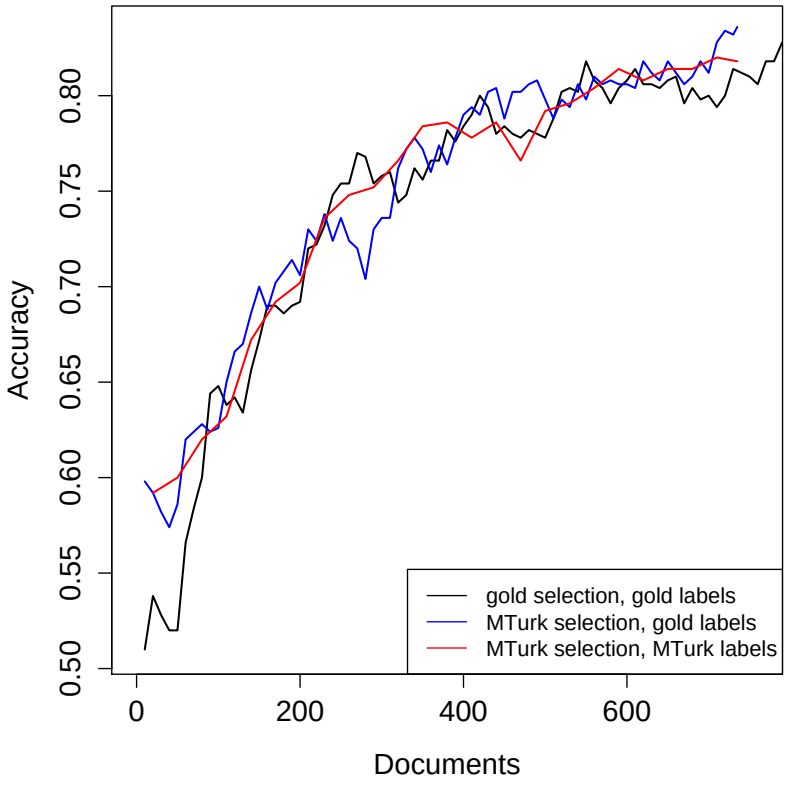


Figure 4.6: Performance using gold annotations (run 1)

MTurk vs. Gold Labels

We showed that 4-voting works considerably better than using single annotations. Conversely, we are also interested in what happens when we use gold annotations instead. There are two possible setups: First, we keep the order of examples as they were selected with MTurk labels (MTurk selection, gold labels). This will show us whether the classification accuracy could be improved with better labels at any point. Second, we run a complete active learning simulation with gold labels, i.e., we also select examples based on gold information (gold selection, gold labels).

The results for these experiments on MTurk run 1 are shown in Figure 4.6. The curves show that the two experiments with gold data and the MTurk run differ only slightly. This is not surprising as we have already seen that 4-voting produces

high-quality annotations. As noted before, we do not expect perfect agreement between human annotators and reviewer ratings.

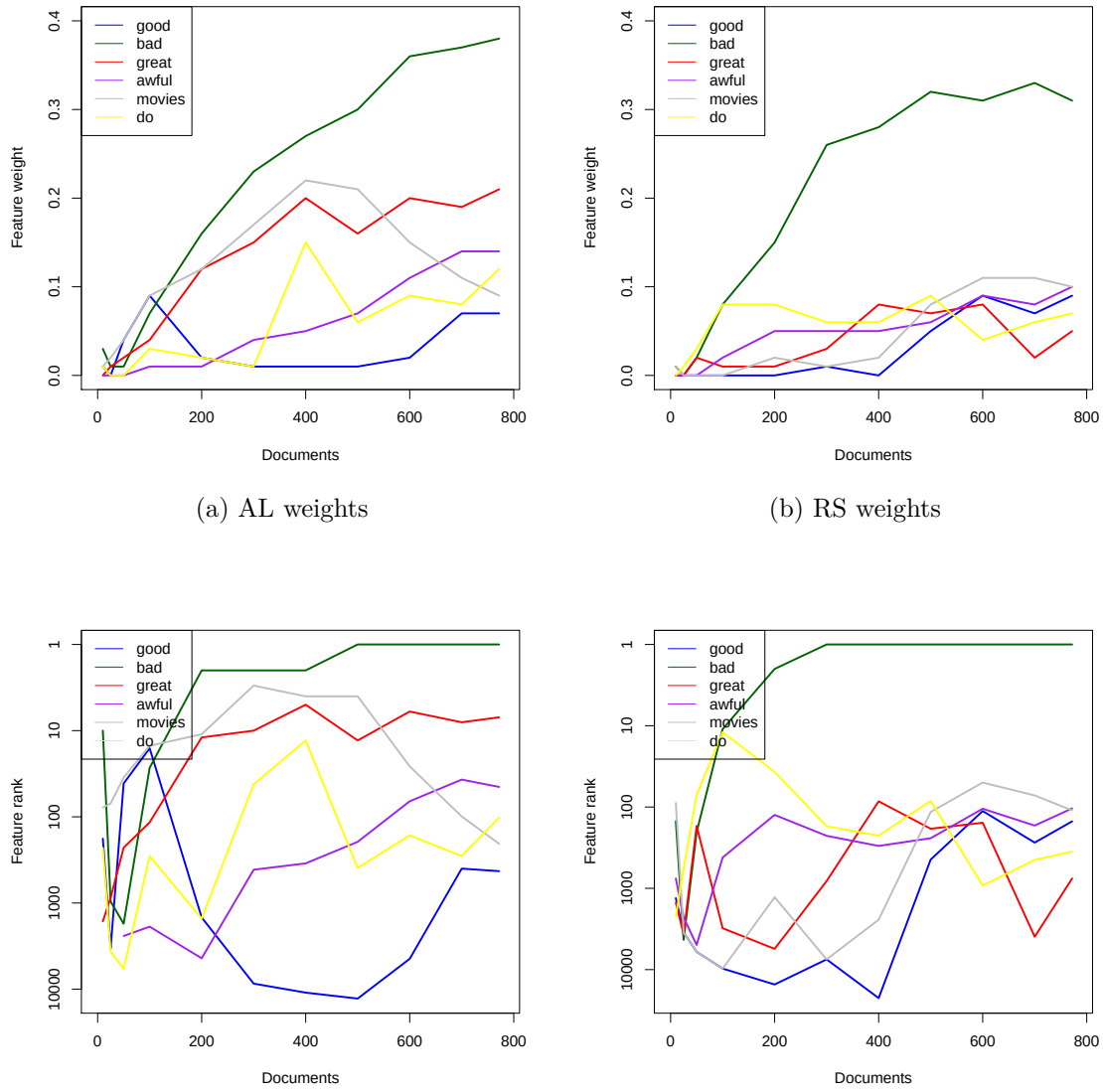
Error Analysis

As noted earlier, we are interested in investigating the effect of active learning on the lexical features. For this reason, we examine the weights the classifier assigns to a selected set of features at different stages of the active learning process (Figure 4.7). We plot for different training set sizes the feature weights for the active learning (Figure 4.7a) and random sampling (Figure 4.7b) classifiers as well as the feature's rank in the decreasingly ordered list of features sorted by the highest absolute value of the weight for both experiments (Figures 4.7c and 4.7d).

We can see that while both methods are able to discover obvious features – such as *bad* which is ranked high early in both active learning and random sampling – we also observe that random sampling fails to discover the feature *great* which is pushed up early in active learning. Conversely, random sampling seems to lead to a fairly conservative assignment of weights while active learning has more high-weighted features. This effect is expected as active learning aims to get labels for documents close to the decision boundary, which leads to a disambiguation of features.

Interestingly, we also observe an overfitting effect with a peak at around 400 documents for active learning. Here, features such as *movies* or even *do* are given some importance by the classifier, possibly due to correlations of these features with one of the classes in the training data. It is at this point in the experiment when we see the highest improvements of active learning over random sampling. Figure 4.4 shows deviation of the two methods starting at around 250 examples which coincides with the point at which several features start to experience rapid ranking changes.

In the final classifier (772 documents), active learning has reached a stable state where features are ranked sensibly. While *bad* is traditionally considered an informative indicator, *good* is not. While active learning briefly considers *good* a helpful feature, it quickly drops it after the 100 document mark.



(a) AL weights

(b) RS weights

(c) AL ranks

(d) RS ranks

Figure 4.7: Feature weights assigned by the AL classifier at different points of the active learning process to some example features. We show both the weight and rank of each feature. Ranks are plotted on a logarithmic scale.

4.4 Related Work

4.4.1 Crowdsourcing

Crowdsourcing has become increasingly popular in NLP due to its ease of use and because it offers a reasonable trade-off between quality and efficiency. Quality control has been an important issue for crowdsourced annotation.

Sentiment analysis was among the first tasks to be tackled through crowdsourcing. This is not surprising as recognizing sentiment is natural to humans, which reduces the required amount of annotator training, compared to more formalized tasks such as syntactic annotation. For example, experiments on emotion labeling were conducted by Snow et al. (2008). Akkaya et al. (2010) use MTurk to perform subjectivity word sense disambiguation. They generally obtain high annotation accuracy, which they further improve by imposing reputation restrictions on the workers. This reduces spam significantly. Taboada et al. (2011) use MTurk to evaluate polarity lexicons. They use voting schemes and agreement checks to improve the quality of the annotations.

While the work summarized above performs crowdsourcing for sentiment tasks, it do not address active learning. It remains unclear from these publications whether a combination of active learning and crowdsourcing is successful.

4.4.2 Active Learning for Sentiment Analysis

Brew et al. (2010) presented one of the first studies on crowdsourcing and active learning for sentiment analysis. They perform sentiment detection in social media. In their experiments, they use a small set of volunteer annotators they recruited for the task. This differs significantly from the completely anonymous setting at MTurk where requesters have little control over who carries out the annotations.

Dasgupta and Ng (2009) propose a combination of different machine learning techniques to improve sentiment classification. They identify difficult examples from the data using spectral clustering. These examples are then passed to humans for annotation. This constitutes a form of active learning as the clustering step performs example selection. The authors show that this process yields a better performance on a small set of labeled documents than traditional, iterative active

learning. The experiments are again simulated.

Li et al. (2012) introduce co-selection, a technique for improving active learning for heavily skewed class-distributions in active learning. Class imbalances are a typical phenomenon in sentiment analysis as often, the positive class is drastically over-represented. The authors find that their method improves active learning in simulated experiments.

There is little research on the role of features in active learning. Settles (2011) presents an interesting approach to active learning where humans can label both features and examples, which is particularly interesting for sentiment applications. The approach combines the two types of information by adjusting the priors of a Bayesian model.

To summarize, we note that there is little work on the combination of active learning and crowdsourcing in general, and none on the performance of this combination in a sentiment analysis task.

4.5 Summary

In this chapter, we presented an approach to sentiment classification using a combination of crowdsourcing and active learning. We used Amazon Mechanical Turk to annotate reviews from the movie corpus. We showed that counter-spam measures and voting strategies are effective for obtaining high-quality polarity annotations.

The selection of documents presented to the workers for annotation was guided through active learning. We showed that with a pool-based uncertainty-sampling strategy, it is possible to achieve significant improvements over random sampling when measuring classification accuracy for equal annotation budgets.

We analyzed the behavior of the active learning process through several experiments. First, we compared the crowdsourcing-based setup to a gold label setup and found no significant difference between the two. Second, we analyzed the clues that are promoted during the selection process. We found that active learning identifies a larger set of interesting clues faster than random selection. However, we also experience periods of overfitting which slow down the active learning process. This shows that while active learning can help to expand the set of clues efficiently, it can also produce misleading clues.

5 Bootstrapping Sentiment Classifiers from WordDocument Graphs

5.1 Introduction

Clues can be annotated on the same polarity scale as documents. Thus, they can be used directly to determine the polarity of a document. This constitutes a form of distant supervision, as we perform supervision through a database of word-level polarity – a polarity lexicon – in order to predict document-level polarity. This type of approach was one among the first for automatic sentiment analysis (e.g., Turney, 2002) and remains popular to date (cf. Taboada et al., 2011). Turney (2002) employs a simple method that involves calculating the average polarity over all clues in a document. A major drawback of this method is that it does not model the interaction between the documents. It only makes use of the information in a polarity lexicon rather than reasoning transitively when clues shared by documents. We argue that the latter would be desirable as documents that share many features are likely to have the same polarity.

We introduce a novel graph-based method for bootstrapping a sentiment classifier from a polarity lexicon that addresses this shortcoming. Graphs offer a way of representing words and documents in a joint formal structure. The structure we introduce, henceforth referred to as *word-document graphs*, is capable of encoding both word-document and word-word relations. In order to predict the polarity of documents, we apply *Personalized PageRank* to the word-document graph, making use of the steady-state distributions of PageRank.

As we have seen earlier, redundancies are a bottleneck in sentiment classification. The distant supervision approach proposed above contributes towards our goal of reducing redundancies among the learned clues. In a polarity lexicon, clues are

annotated out of context, i.e., by type rather than by token. Thus, we annotate each clue only once, which is intuitively more efficient than increasing coverage of clues through active learning on the document level. Type-level annotation is motivated by the same independence assumptions we already made on the document level: Each word has the same polarity regardless of its context.

There are known issues related to lexicon-based distant supervision. First, related to the aforementioned independence assumption, prior polarity not correct across all contexts/domains (Kanayama and Nasukawa, 2006). Second, not all clues might be covered by either the lexicon or the word-document graph. Therefore, we propose to perform an additional bootstrapping step in which we use the document labels generated through distant supervision to train a supervised classifier. This model will have access to all unigram features in the text. Thus, it can use a larger feature space than the distant supervision model and can assign weights the clues by indirectly making use of the polarity lexicon.

The rest of the chapter is structured as follows. We first introduce the necessary background in Section 5.2. Next, we give an overview of novel methods, introducing word-document graphs and *Polarity PageRank* (PPR), a new semi-supervised sentiment classifier that integrates lexicon induction with document classification (Section 5.3). We then conduct experiments with PPR in Section 5.4. We show that PPR yields improved polarity classification performance over average polarity on an English reference review corpus. Classification accuracy on documents can be further improved by bootstrapping a statistical classifier from the PPR output.

5.2 Background

5.2.1 Polarity Induction with Word Graphs

The induction of polarity on a word graph was first introduced by Hatzivassiloglou and McKeown (1997). In their approach, the authors construct a syntactic dependency graph (cf. Section 2.5) of conjunction relations between words. Formally, a *word graph* is an undirected graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ where the nodes $\mathcal{V}$ represent words, the edges $\mathcal{E}$ represent an semantic relation between the words, and $\mathbf{W}$ represents the strength of the relation. Data for the word graph used by Hatzivassiloglou

and McKeown (1997) is collected by counting conjunctions in a text corpus. The edges in the graph are weighted by the frequencies of these occurrences. In the original paper, words coordinated with *and* are assumed to share polarities while words coordinated with *but* are expected to have opposing polarities. Consider the following examples:

(5.1) *nice and good*

(5.2) **lovely and despicable*

(5.3) *awful but funny*

(5.4) **lovely but entertaining*

(5.5) ?*good and bad*

The phrases in 5.1 and 5.3 are examples of what we would expect to find in actual data. In contrast, example 5.4 is implausible. Examples 5.2 and 5.5 are fringe cases. It is of course possible to use *and* conjunctions for contrastive purposes, and thus we have to expect cases with words of opposing polarities. However, this phenomenon is expected to be rare and therefore treated as noise.

We intend to apply PageRank, a probabilistic Markov chain random walk approach, to the graph. The random walk is guided by transition probabilities, and thus, we cannot make use of negative links easily. For this reason, we restrict ourselves to *and* edges in the graph. An example for such a graph is provided in Figure 5.1. The figure shows the neighborhood of distance 1 for the word *warm*.

This type of graph is an instance of a syntactic dependency graph. However, it is not necessary to actually use syntactic parses for finding conjunctions. Indeed, using parses may actually lead to increased errors because coordination relations are commonly among the most difficult syntactic structures for parsers to analyze correctly (McDonald, 2006). Alternatively, conjunction relations may be detected based on part-of-speech patterns. We can specify such a pattern through a regular expression for which we search matches in the sequence of part-of-speech tags in a corpus. We use the following expression to extract adjective conjunctions from the corpus:

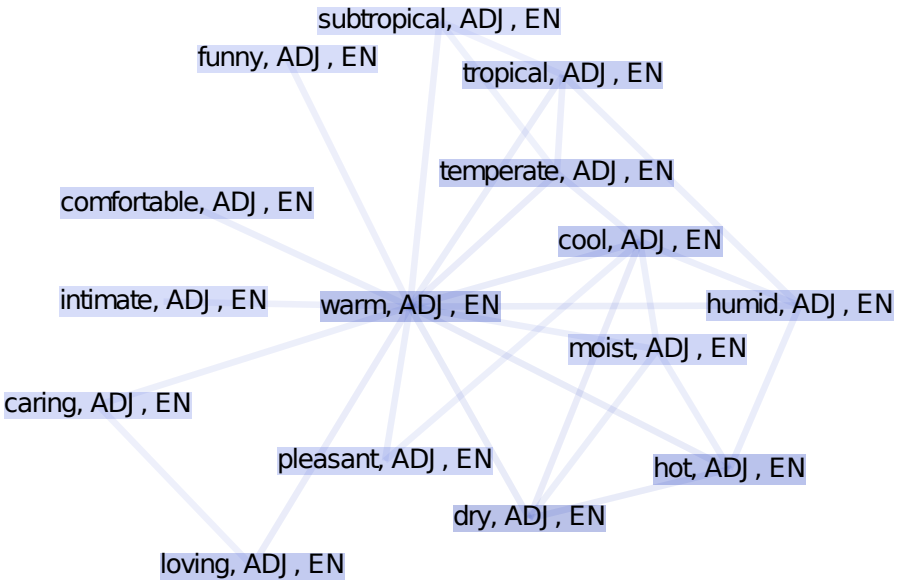


Figure 5.1: Word graph – 1-neighborhood of the word *warm*

```
[pos ="JJ"] ([pos = ","] [pos = "JJ"])*([pos = ","]?
"and" [pos ="JJ"])+
```

The pattern extracts conjunctions with *and* as well as possibly preceding words that are enumerated by commas. It is of course applicable to any other part of speech by replacing the adjective tag `JJ` by the respective tag. It is important that all matched words have the same part of speech as otherwise, the pattern has a tendency to overgenerate, for example by matching borders of larger coordinated units such as coordinated verbal phrases (e.g., we would extract *home and took* from *he went home and took a nap*).

5.2.2 Average Polarity

One of the earliest and most popular methods for using clues to determine the polarity of a document is *average polarity* which we adopt as our baseline method. It was applied first by Turney (2002). Given a polarity lexicon of clues, we simply average the polarity values of all clues that occur in the document. The average

polarity pol of all words w in a document d is

$$\text{pol}(d) = \sum_{w \in d} \frac{\text{pol}_l(w)}{|d|},$$

where pol_l is the lexical polarity taken from the polarity lexicon. We assume that the lexicon has polarity ratings in $[-1, 1]$. We can then classify the document d by checking whether its score is positive or negative:

$$y_d = \begin{cases} \text{positive} & \text{if } \text{pol}(d) \geq 0 \\ \text{negative} & \text{else} \end{cases} \tag{5.6}$$

The classification confidence c of a document d is assessed through $c(d) = |pol(d)|$. Note that we still assign a class to documents where pol(d) = 0. This classification divides the set of documents into two partitions, i.e., each document is either positive or negative. This decision follows related work (e.g, Hassan and Radev, 2010) and incorporates the majority baseline into the model. This way, nodes for which PPR cannot make a prediction are assigned the majority class which is more likely to be correct. The intuition behind this is similar to the one of class priors in probabilistic models.

Average polarity has several drawbacks. First, it is very sensitive to incorrect entries in the polarity dictionary. For example, the Wilson et al. (2005b) lexicon lists *need* as a negative word. This might be appropriate in some contexts, but it is arguably not the prior polarity of the word. Second, there is no interaction between the documents, i.e., the occurrence of clues in several documents is not taken into account in the classification decision. Third, it is not trivial to combine this method with document-level supervision.

5.3 Methods

5.3.1 Polarity PageRank

To infer the polarity of unlabeled nodes in a graph of similarity relations between words, we adopt a random-walk-based approach. For that, we make use of Per-

sonalized PageRank for polarity, calling the resulting algorithm *Polarity PageRank* (PPR). We assume that we know the polarities of some nodes, for example from an existing polarity lexicon. These nodes make up the *seed sets* $\mathcal{S}^+$, the set of positive nodes, and $\mathcal{S}^-$, the set of negative nodes. The idea behind PPR is to use these seed sets for two independent runs (*positive* and *negative*) of Personalized PageRank. In the *positive* (resp., *negative*) run, the teleportation vector $\mathbf{t}^+$ (resp., $\mathbf{t}^-$) is set as defined in Equation 2.3 so as to give high weights to nodes whose polarity is known to be *positive* (resp., *negative*). We apply PPR to a dependency graph, so the final transition matrix will be computed from the adjacency matrix of the dependency graph and the teleportation vector following Equation 2.2. Applying PageRank yields a steady state vector for each class, i.e., $\mathbf{r}^+$ and $\mathbf{r}^-$ in our case. Each vector will contain high probabilities for those nodes that are well-connected to the respective seed set, which signifies that they important to the respective classes. From these vectors, we can calculate the polarity of a node i (similarly to the previously defined Equation 5.6) as

$$y_i = \begin{cases} \text{positive} & \text{if } r_i^+ \geq r_i^- \\ \text{negative} & \text{else.} \end{cases} \tag{5.7}$$

The assignment of a class for nodes where $r_i^+ = r_i^-$ follows the same principle we introduced for average polarity in Section 5.2.2.

PPR bears some resemblance to the method of Hassan and Radev (2010) (cf. Section 5.5). However, it is simpler, using standard eigenvector computations that are available in any numerical software library, and it is also more efficient, avoiding the need for expensive Monte Carlo sampling. Computational simplicity and efficiency are of particular importance as the graphs we will work with are large due to the fact that they will represent both words and documents.

There are some theoretical concerns regarding the application of PageRank in this particular type of setup. It has been recognized that applying Personalized PageRank to undirected graphs can lead to close-to uniform stationary distributions (Grolmusz, 2012). This is most problematic in cases where teleportation vectors are dense and uniform. For sparse vectors where only a small number of non-zero entries have uniform weights, as it is the case in our setup, the stationary

distribution is expected to deviate from the uniform distribution, which is sufficient for classifying nodes.

5.3.2 Document Classification with Polarity PageRank

The main focus in this chapter is to improve the integration of clues into a statistical model for document classification. This approach can be viewed as a form of distant supervision as we do not annotate documents directly but rather rely on a database of clues. Common methods in this scenario are variations of average polarity, use lexical scores as hard-coded features for a more sophisticated classifier (e.g., Melville et al., 2009), or apply a mixture of both approaches (e.g., He, 2010). We will discuss these approaches in more detail in Section 5.5.

The main drawback of these approaches is that they do not offer a principled joint model of word and document polarity. To this end, we propose a graph framework that integrates polarity induction and lexicon application into a unified step. In this model, the occurrence of a word in a document is represented as an edge in the graph, following the definition of an occurrence network in Section 2.5. This way of formalizing the problem is not unlike many information retrieval methods that also view words and documents as the same formal object (e.g., Turtle and Croft, 1991). A similar framework has been used for sentiment aspect analysis by Zhang et al. (2010).

We define a joint graph structure, extending the concept of word graphs (see Section 5.2.1) to include documents. We introduce the *word-document graph* here, an undirected graph which contains both *word-word* and *word-document* edges. We adopt the word-word edges from the word graph. In addition, word-document edges are introduced between each document node d_j and all terms t_i that occur in it (i.e., $(i, j) \in \mathcal{E}$ for all $t_i \in d_j$). The degree of association between a term and a document is given by their normalized term frequency tf (Salton and McGill, 1986),

$$\mathrm{tf}_{ij} = \frac{n_{ij}}{\sum_k n_{kj}},$$

where n_{ij} is the occurrence count of term t_i in document d_j.

In contrast to the standard setup of PageRank in document retrieval, the documents are not linked directly to each other. Instead, document nodes are linked

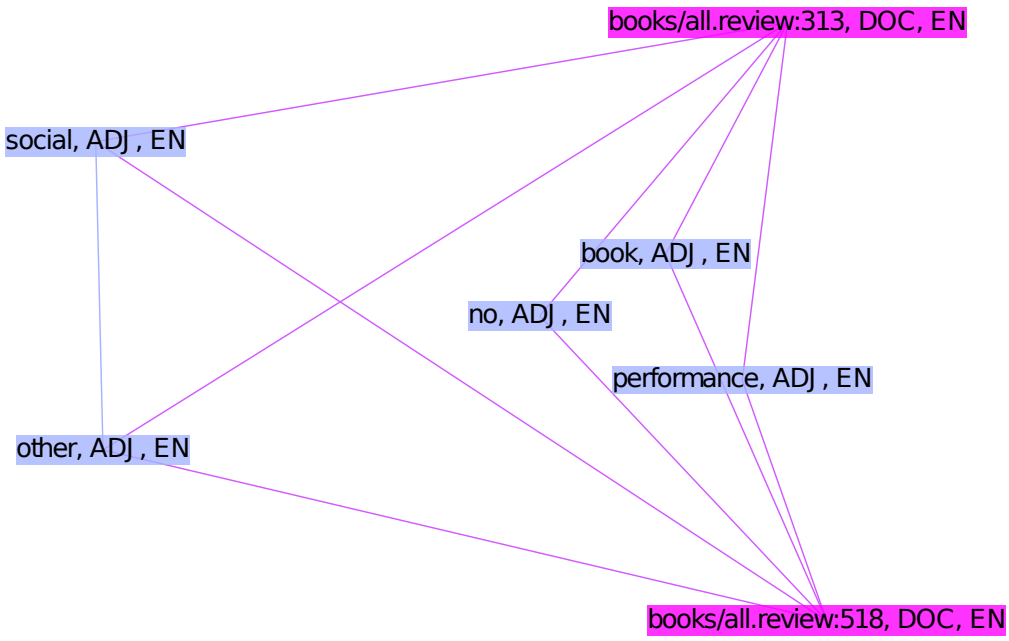


Figure 5.2: Word-document graph. Word nodes and word-word edges are shown in blue, document nodes and word-document edges in magenta.

only to word nodes. As word nodes can be linked to each other, the relationships between documents are defined through the relationships of their terms. The relations necessary for these links do not have to be obtained from the documents in the graph but can be gathered from external sources as well, like in our case through a word graph of conjunctions. Figure 5.2 contains an example of a word-document graph, consisting of 5 word nodes (blue) and 2 document nodes (magenta). By including documents as nodes in the graph, the documents themselves can be labeled with polarity using PPR. We simply use Equation 5.7 for the joint graph. Confidence for a node i is assessed by the absolute log ratio of r_i^+ and r_i^-,

$$\text{conf}(i) = \left|\log \frac{r_i^+}{r_i^-}\right| = \left|\log r_i^+ - \log r_i^-\right|.$$

The word-document graph framework is flexible and can be easily extended. As an example, word-document graphs could be made bilingual. Given word-document graph in two languages, the graphs can be combined by adding links

from a standard bilingual dictionary that translates between the languages (cf. Scheible, 2010). A further possible modification is the introduction of multiple edge types (cf. Scheible et al., 2010).

5.3.3 Bootstrapping

The word-document graph model with PPR offers the advantage of having a joint model of words and documents. However, as PPR is transductive, we are not making use of some of the elements of statistical learning. Most notably, the ability to weight features based on their importance to a class is absent. This motivates the use of bootstrapping (cf. Section 2.3) where we train a statistical classifier using an automatically labeled pool of examples, exploiting confidence estimates that have been made for each of the predictions.

We propose the following process. Given a set of unlabeled documents, along with a polarity lexicon and a word graph, we first predict document labels using PPR. We then use the confidence measure we defined for PPR to select the most confidently labeled documents. Using these documents, we then self-train a statistical classifier and apply it to the set of documents. The motivation behind this approach is that we can produce a base classifier and label a small set of documents with the highest confidence using a polarity lexicon. These documents and their annotations are training input for a supervised classifier that learns weights for all clues in the data, not just those that are represented in the word graph. In addition, this approach makes it possible to use more complex features, such as n-grams.

5.4 Experiments

5.4.1 Experimental Setup

Experiments are carried out on the Multi-Domain Sentiment Dataset (MDS) Blitzer et al. (2007). This dataset is described in more detail in Appendix A.2.1. Testing the method on different domains is interesting domain effects can have a significant influence on the performance of the method. For each category, all available

reviews (positive, negative, and unlabeled) are merged into a joint collection. Since we do not use the available document labels for classification, we can evaluate on the complete dataset. We create an individual word-document graph for each domain, as we found that domain interaction effects can lead to significantly lower accuracy.

We construct an English word graph by extracting adjectives that occurred in conjunctions with *and* from the English Wikipedia as described in Section 5.2.1, using a Wikipedia dump downloaded on 2/1/2009. The edges for the resulting graph are weighted by conjunction occurrence counts. We use Wilson et al. (2005b)'s English polarity lexicon. We can make use of only those entries that are represented as (word) nodes in the graph, as not all words are covered by Wikipedia or the MDS. Thus, we use only part of the lexicon, around 2000 words.

We run PPR until convergence on the dataset, taking around 70 iterations for each run. We set the PPR teleportation parameter $\alpha = 0.85$, a standard setting in information retrieval. For bootstrapping, we use a maximum entropy (MaxEnt) model (Stanford classifier implementation, see Appendix A.1.1) with bag-of-words features. We perform a single self-training iteration based on the PPR output, taking the 15% most confidently classified documents of the collection as training data, using a class-balanced sample. We then iteratively select (without replacement) the best 1% of the remaining documents. We use the pre-processed version of the MDS, thus the features available to the classifier are the unigrams and bigrams as extracted by the original creators of the dataset.

We try two variations of Polarity PageRank. In the first version, we label the documents based on the teleportation vector constructed from the given polarity lexicon with a single PageRank run for each class (referred to as PPR). The second version first calculates the positive and negative eigenvector on the word graph in a first PPR run and then uses them as teleportation vectors on the word-document graph in a second PPR run (referred to as PPR+).

5.4.2 Experiments and Results

The top section of Table 7.1 shows the accuracy of the average polarity (AP) and Polarity PageRank (PPR, PPR+) base classifiers before bootstrapping on

Method	**Books**	**DVD**	**Electronics**	**Kitchen**	***Mean***
AP (base)	51.2	58.5	57.1	56.6	55.9
PPR (base)	60.3†	63.4†	62.0†	62.0†	61.9
PPR+ (base)	**68.7**†	**67.7**†	**67.1**†	**67.6**†	**67.8**
AP (MaxEnt)	70.4	69.6	74.1	75.7	72.5
PPR (MaxEnt)	69.3	71.0	74.2	77.6	73.0
PPR+ (MaxEnt)	**71.1**	**72.5**†	**76.7**†	**80.3**†	**75.2**

Table 5.1: Classifier accuracies. † denotes a significant improvement with $p < 0.05$ over the AP result of the same section.

the different corpora splits for each domain. The overall accuracies in the last column are means over the four domains. We apply the approximate randomization significance test to the results and let † denote a significant improvement with $p < 0.05$ over the AP result of the same section. We see significant improvements over baseline for all domains using PPR instead of AP, and another significant boost when using PPR+. The latter result comes as no surprise as the intermediary PPR step serves to expand the lexicon, enabling PPR+ to use significantly more input data.

Based on these results we apply bootstrapping and self-training as described above. The accuracies of the resulting classifiers are listed in the bottom section of Table 5.1. The improvements of PPR maximum entropy models over AP models is smaller than the margin of the base models, which is due to the availability of more features to the classifiers. We see significant improvements over the AP MaxEnt baseline only when using PPR+, except for the books domain where neither PPR nor PPR+ yield a significant improvement. This result can be explained by looking at the average document length across domains (cf. Appendix A.2.1). Documents are the longest on average for the books domain and the shortest for the kitchen domain. This means that the AP classifier can make better confidence estimates as the overall number of clues in a document is larger. If a decision is made with high confidence, it is most likely correct. Therefore, the margin between AP and PPR is

smaller for books than for kitchen documents. In our experiments, we found that subsequent self-training iterations do not help to improve the results significantly. One reason for that mirrors an effect we already observed in Chapter 4: It is difficult to make improvements after learning a basic set of good lexical indicators.

The books domain is the most difficult domain overall, followed closely by the DVD domain. One reason for this is that these domains contain confounding clues, such as plot descriptions, which are wrongly recognized as polarity clues by the classifier. A separate step detecting clues that are non-relevant to sentiment analysis may be desirable. We will investigate such an approach in the following chapter.

Error analysis also shows that the bootstrapping step manages to correct the polarity of some document nodes which are not well connected in the graph and therefore are difficult or impossible to classify with PPR. This motivates a deeper analysis of the graph, which we present in the following section.

5.4.3 Analysis of Word-Document Graphs

In this section, we provide some analysis of the graph's properties by means of visualization. First, we visualize the complete Wikipedia word graph with the nodes organized through a force-directed layout.[13] In Figure 5.3, we notice that the graph consists of several disconnected subgraphs. There are many small connected components and singleton nodes that are unreachable from the largest connected component, shown at the edges of the figure. In the largest connected component, the force-directed layout produces several clusters. To analyze them, we filter the graph and look only at nodes with a degree of at least 25 (Figure 5.4). Using node labels, we can identify the topics of three of the four large clusters. The top-left cluster contains adjectives describing origin and nationality (e.g., *French*, *Italian*, and *European*). Nationality terms are strongly connected among each other and rarely occur in conjunction with other words. This cluster is connected to the rest of the graph through certain hub nodes that are less specific, such as *foreign* and *international*. The top-right cluster contains adjectives describing temporal or spatial relations (e.g., *first*, *final*, and *early*). It is rather small as such expressions

[13] We use the implementation in Gephi, available at `https://marketplace.gephi.org/plugin/openord-layout/`

are either rare or not productive. The bottom-left cluster contains words related to society and professions (e.g., *political*, *cultural*, and *professional*). The bottom-right cluster contains the remainder of the adjectives, which are mostly qualitative (e.g., *long*, *powerful*, *popular*, and *new*). Presumably, these adjectives are the most interesting for sentiment classification. The two most obvious sentiment indicators *good* and *bad* are both part of this cluster as well. The four clusters are interconnected through hub nodes such as *other* and *many*. Direct connections are more rare, as conjunctions of words that are less semantically related are implausible. This supports the conjunction hypothesis of Section 5.2.1.

We next look at an example from the bottom-right cluster in more detail. We show the neighborhood of distance 1 for the word *powerful*, a central word in the cluster, in Figure 5.5. We can see that even this small neighborhood is well-connected. Many of the words in this subgraph are positive according to the polarity lexicon, such as *successful* or *beautiful*. The neighborhood graph also contains words that could be considered ambiguous, such as *expensive* whose polarity depends on the context in which it is used. There are also direct connections from *powerful* to negative words such as *evil* and *corrupt*. This shows once more that the assumptions made in Section 5.2.1 are not universally true.

Figure 5.6 shows the word-document graph for the *books* domain. We can see that the majority of the nodes in the graph is part of a large connected component with two obvious main clusters. In addition, we have many unreachable nodes and connected components, some of which contain document nodes. These nodes are difficult to label through PPR as they can only be classified if at least one of the words they are connected to is in the seed set.

The two main clusters of the largest connected component are straightforward to interpret: The cluster in the bottom part of the figure contains mostly word-word relations, shown in blue. The cluster on top contains mostly word-document relations, shown in magenta. In the top cluster, we also find word nodes that share no edge with the main word graph, i.e., words that occur in documents but not in any conjunction. We also see that the two clusters are sparsely connected through direct word-document edges (magenta edges between the clusters). Examining these edges shows that the document nodes mostly connect to the cluster of qualitative adjectives described above.

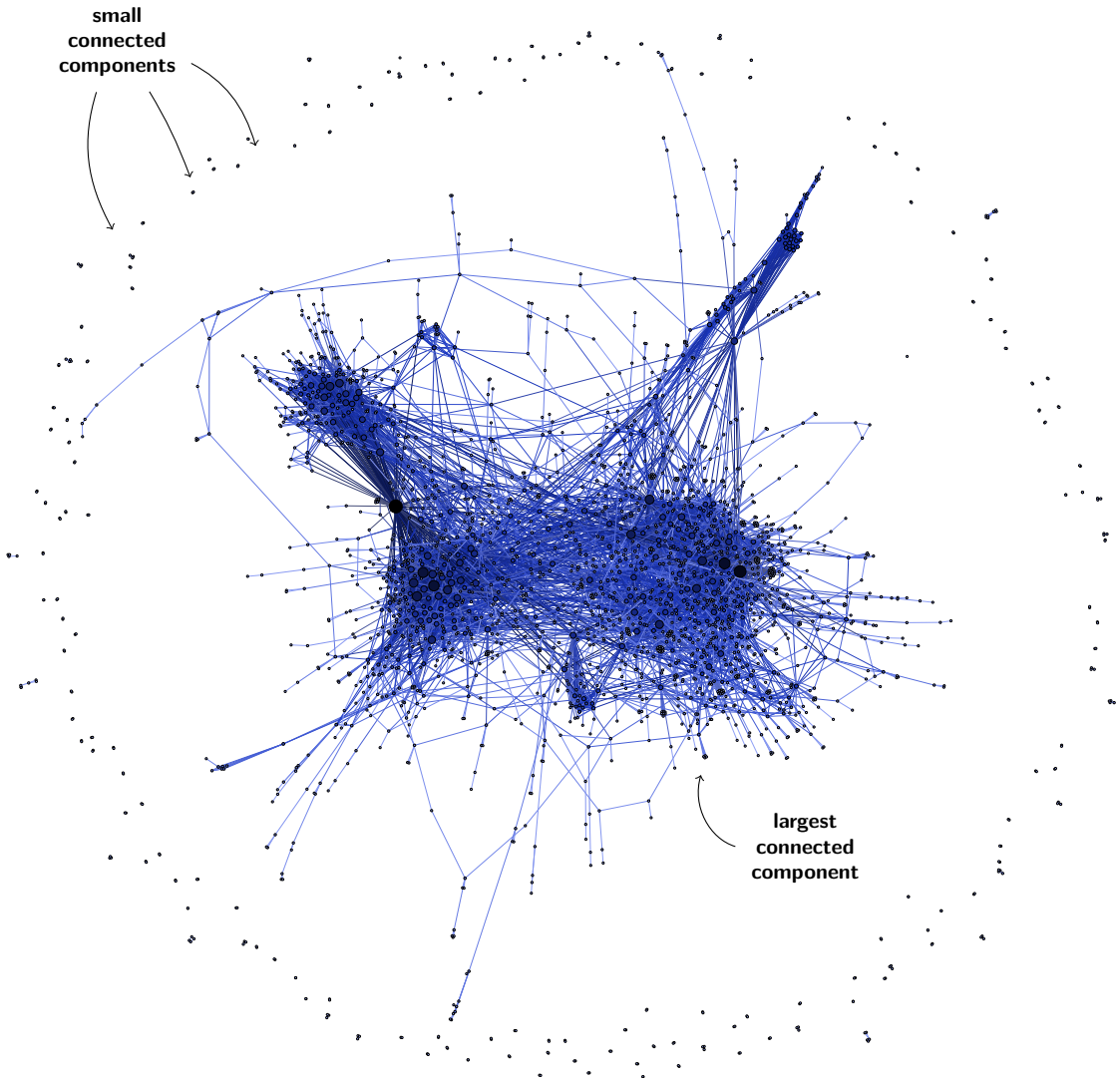


Figure 5.3: Complete Wikipedia word graph. Color intensity and node size visualize the degree of each node.

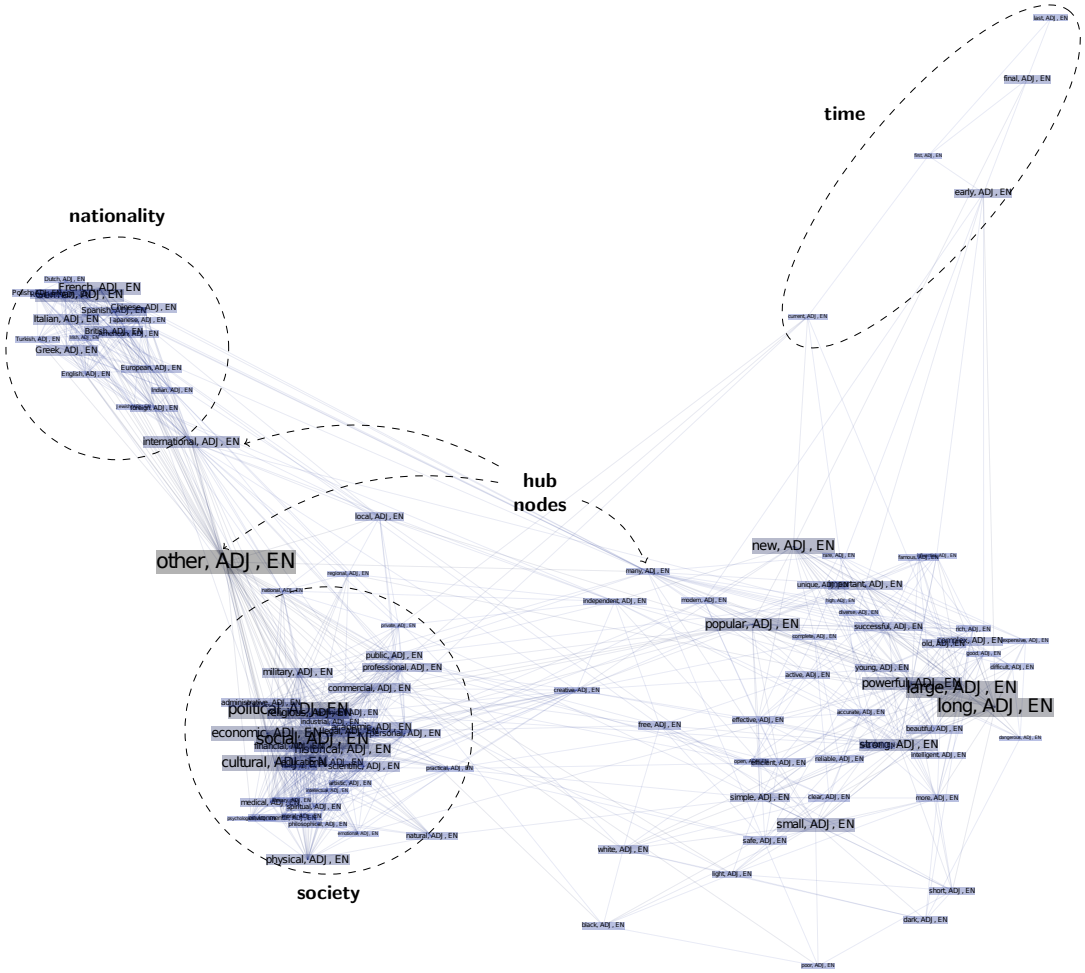


Figure 5.4: Wikipedia word graph with word labels. We show all nodes with a degree of at least 25. Manually identified coherent clusters circled.

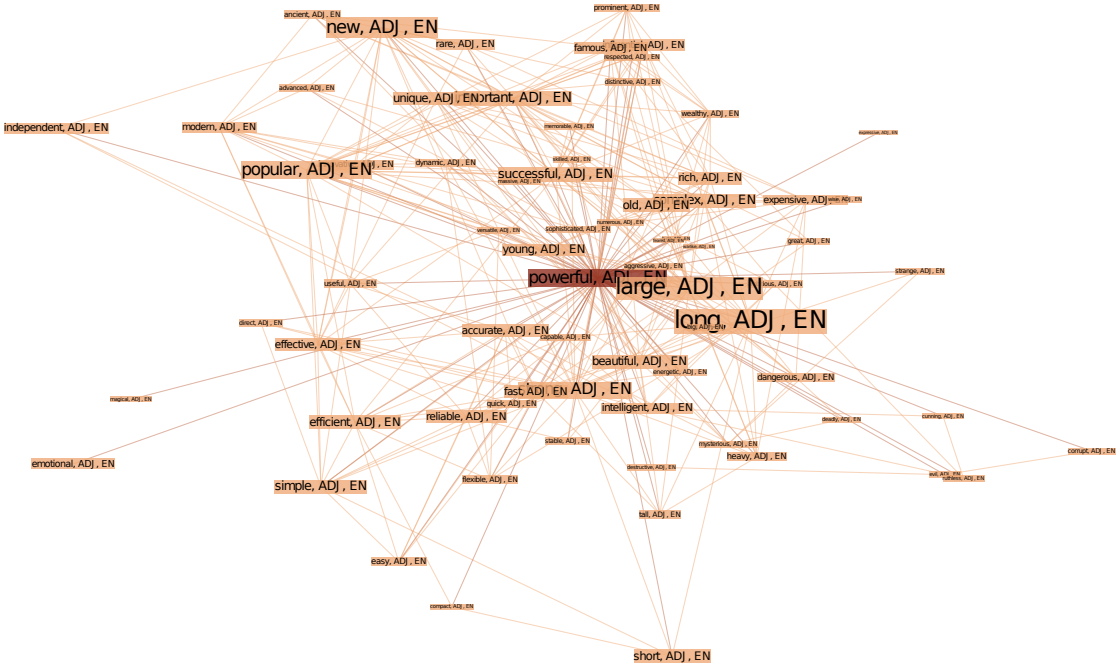


Figure 5.5: 1-neighborhood of *powerful*. Color intensity denotes connection strength to *powerful*.

These observations suggest that further pre-processing of the graph, e.g., removing unnecessary or misleading content, could potentially help to improve sentiment analysis.

5.5 Related Work

5.5.1 Lexical Knowledge in Document Classification

Leveraging lexical clue knowledge for review classification has been subject of several related publications. Turney (2002) induces a polarity lexicon by measuring the association of terms with a set of seed words whose polarity is known. The resulting lexicon is used for classifying reviews by calculating the average polarity of each document. Turney concludes that these averages are highly correlated to the actual polarity of the documents.

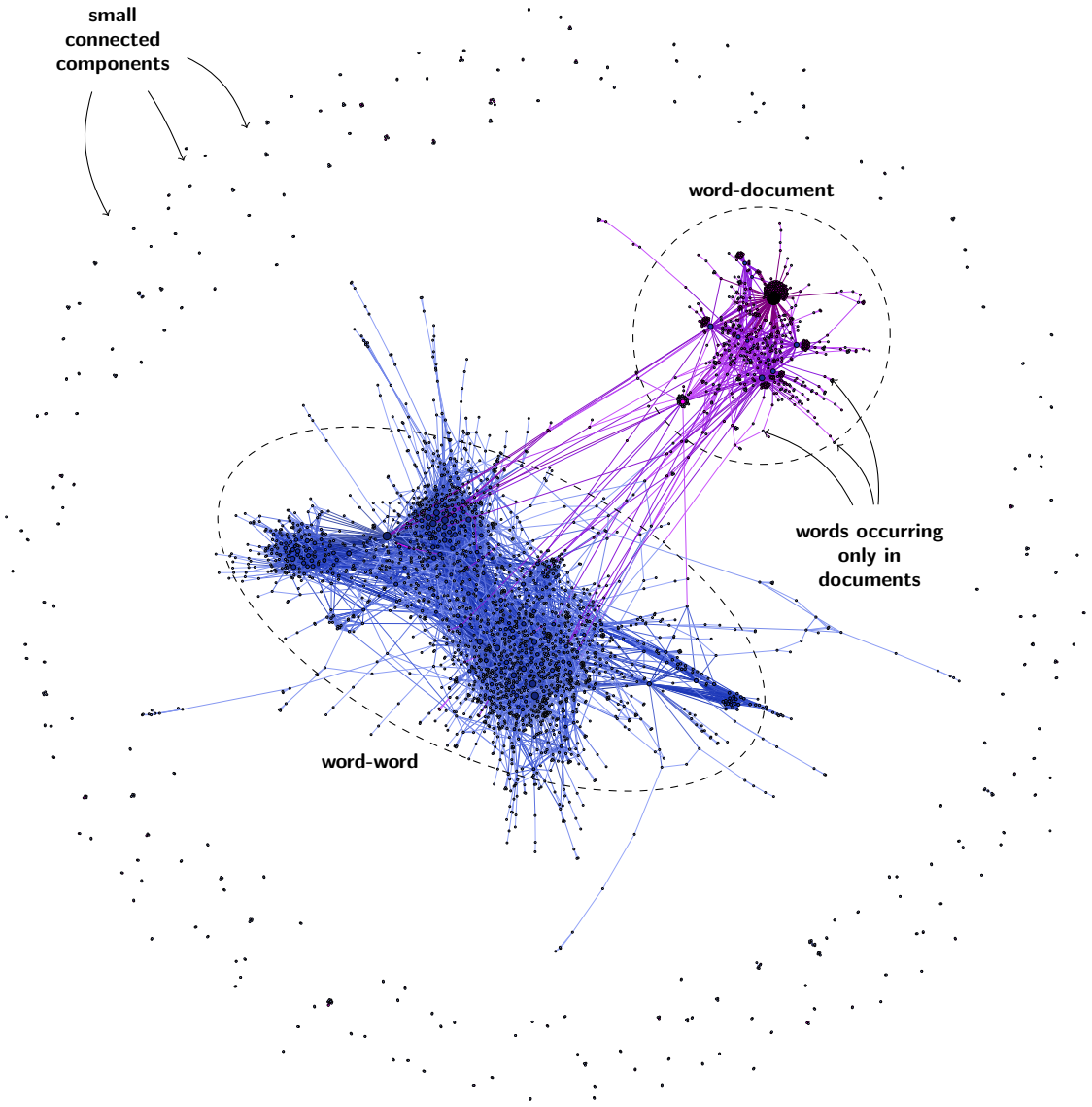


Figure 5.6: Complete word-document-graph for the *books* domain. Word nodes and word-word edges are shown in blue, document nodes and word-document edges in magenta. Major regions of the graph are circled.

He (2010) presents a self-training approach for review classification. The importance of each lexicon item is taken into account and is estimated from the unlabeled texts. This leads to an increase of accuracy in review classification. The focus of He's work is on correctly estimating the importance of each feature for sentiment classification.

Settles (2011) introduces a Bayesian model using Dirichlet priors for feature labeling. The model aims at improving active learning for document classification tasks, including sentiment analysis. Clue information can be introduced by modifying the priors. The advantage of this model is that it is theoretically well-motivated. In its current form, it limits the user to a Naive Bayes model which might not be preferred for all applications (cf. for example Pang et al., 2002, who show that MaxEnt performs better on movie review classification).

5.5.2 Random Walk Methods for Polarity Induction

Zhang et al. (2010) perform aspect extraction with the HITS algorithm, a random walk method closely related to PageRank. Their graph setup is based on dependency graphs of predicate-argument relations that encode the relation of aspects and opinion words. They evaluate their approach on a 4-domain dataset which is not publicly available.

Random walk methods have been applied successfully for polarity induction on word graphs. Hassan and Radev (2010) induce a polarity lexicon by constructing a graph where edges represent taxonomic relations from WordNet, such as hypernymy. Polarity is propagated on this graph using a random walk model that handles positive and negative words separately. Their approach outperforms the previous state of the art method for polarity clue induction. In contrast, we adopt document classification accuracy as our evaluation metric.

5.5.3 Bootstrapping

Bootstrapping was shown to be an effective tool in sentiment analysis previously. Wiebe and Riloff (2005) introduce a bootstrapping approach for subjectivity classification that learns patterns of subjectivity clues from unannotated texts. These

clues serve as a source for a Naive Bayes classifier that produces additional high-confidence input for the pattern learner. Initial rules need to be hand-crafted which requires linguistic expert knowledge about a language.

5.5.4 Conclusion

In comparison to the related document classification approaches summarized above, our method has two advantages. First, our initial classifier uses an efficient graph-framework with PageRank which is well-researched. Second, the bootstrapping step introduces supervised classification techniques, weighting clues in the process, leveraging information from the entire feature space. Third, our approach requires little knowledge about a language, therefore eliminating the need for linguistic expert knowledge.

5.6 Summary

In this chapter, we introduced Polarity PageRank, a new semi-supervised sentiment classifier that integrates lexicon induction with document classification in one unified graph-theoretic formalism. We were able to show that Polarity PageRank-based document classification improved results over document classification with average polarity. These accuracy improvements translate to increased performance of subsequently trained maximum entropy classifiers across all domains. We showed that through bootstrapping, we can take advantage of the entire feature space available. We also observed a well-known shortcoming of the clue model. All clues are treated as equally relevant independently of context, which misleads the classifier. We will further address this issue in Chapter 6.

Possible future work consists of including more sophisticated features in the graph, e.g., negation-based features or bigrams. It is an open question if the PPR would be as accurate as bootstrapping if the full feature set could be made available to PPR. In addition, the approach can be extended to multilingual sources by using multiple word-document graphs and a bilingual dictionary.

6 Sentiment Relevance

6.1 Introduction

It is generally recognized in sentiment analysis that only a subset of the content of a document contributes to the sentiment it conveys (cf. Liu, 2010). We made this observation ourselves. In the previous chapter, we found the counter-intuitive result that our classifier performs better for shorter documents. We attribute this phenomenon to topic effects that introduce misleading clues. For this reason, we believe that automatically identifying such distracting content could be beneficial to downstream sentiment analysis tasks. We address the concept of *sentiment relevance* in this chapter which formalizes this issue.

Filtering unwanted contexts is an established pre-processing step in sentiment analysis (Liu, 2012). To identify such content, some authors make use of the distinction between the categories *subjective* and *objective* (see Section 3.1.2). Subjective statements refer to the internal state of mind of a person, which cannot be observed. In contrast, objective statements can be verified by observing and checking reality. Some sentiment analysis systems filter out objective language and predict sentiment based on subjective language only because objective statements do not directly reveal sentiment. Even though the categories subjective/objective are well-established in philosophy, we argue that they are not optimal for context filtering We instead introduce the notion of *sentiment relevance* (SR). A sentence or linguistic expression is sentiment relevant if it contains information about the sentiment the document conveys; it is *sentiment nonrelevant* (SNR) otherwise.

Although there is overlap between the two notions, they are different. Consider the following examples for subjective and objective sentences:

(6.1) *Bruce Banner, a genetics researcher with a tragic past, suffers a horrible accident.*

(6.2) *The movie won a Golden Globe for best foreign film and an Oscar.*

Sentence 6.1 is subjective because assessments like *tragic past* and *horrible accident* are subjective to the reader and writer. Sentence 6.2 is objective since we can check the truth of the statement, i.e., whether the movie did win the award. However, even though sentence 6.1 has negative subjective content, it is not sentiment relevant because it is about the plot of the movie and can appear in a glowingly positive review. Conversely, sentence 6.2 contributes to the positive opinion expressed by the author through a factual statement. Subjectivity and sentiment relevance are two distinct concepts that do not imply each other: Generally, objective sentences can be sentiment relevant, and subjective sentences can be sentiment nonrelevant. Below, we demonstrate empirically that subjectivity and sentiment relevance differ.

Ideally, we would like to have at our disposal a large annotated training set for our new concept of sentiment relevance. Unfortunately, this resource does not yet exist. For this reason, we investigate two semi-supervised approaches to sentiment relevance classification that do not require sentiment relevance -labeled data. The first approach uses distant supervision (cf. Section 2.3). We create an initial set of distantly labeled examples based on domain-specific metadata that we extract from a public database and show that this improved performance by 5.8% $\bar{F}_1$ compared to a clue model baseline. The second approach uses transfer learning (cf. Section 2.3). We show that transfer learning improves $\bar{F}_1$ by 12.6% for sentiment relevance classification when we use a feature representation based on lexical taxonomies that supports knowledge transfer.

In the following, we analyze sentiment relevance on the *sentence* level. Our motivation is two-fold. On the one hand, sentences are a low-level linguistic structure where sentiment relevance manifests itself in a relatively unambiguous manner. On the other hand, it is easy to segment texts into sentences as opposed to other linguistic structures such as phrases with identifiable sentiment relevance. However, sentiment relevance is also a discourse phenomenon: authors tend to structure documents into sentiment relevant passages and sentiment nonrelevant passages. To impose this discourse constraint, we employ a sequence model. We represent each document as a graph of sentences and apply minimum cut.

The rest of the chapter is structured as follows. First, we describe the sentiment relevance annotation process and show experimentally how sentiment relevance

differs from subjectivity (Section 6.2). Section 6.3 introduces the methods applied in this chapter and the features we extract. Then, we turn to the description and results of our experiments on distant supervision (Section 6.4) and transfer learning (Section 6.5). Finally, in Section 6.6, we review previous work and show how sentiment relevance relates to it.

6.2 Exploring Sentiment Relevance

Since sentiment relevance is a novel concept, we first need to create an annotated corpus in order to conduct computational experiments. In this section, we begin by describing the annotation process including an agreement study. Then, we conduct experiments to compare sentiment relevance to the most closely related concept, subjectivity. We focus on sentiment relevance in the movie domain for several reasons. First, it is a relatively closed domain with a fixed set of aspects that has been the focus of much prior research. Second, we found the quality of movie reviews to be generally higher than the one of product reviews. Third, non-relevant content occurs frequently in movie reviews as they tend feature long descriptions of plots. This leads to a less skewed class distribution.

6.2.1 Sentiment Relevance Corpus

To create a sentiment-relevance-annotated corpus, the *SR corpus*, we randomly selected 125 documents from the movie corpus (cf. Appendix A.2.1), using the texts from the raw HTML files since the processed version does not have capitalization.

Two human judges annotated the sentences for sentiment relevance, using the labels `SR` and `SNR`. If no decision can be made because a sentence contains both sentiment relevant and sentiment nonrelevant linguistic material, it is marked as `uncertain`. We excluded 360 sentences that were labeled uncertain from the evaluation. Annotator agreement is discussed in Section 6.2.2. In total, the SR corpus contains 2759 sentiment relevant and 728 sentiment nonrelevant sentences. Figure 6.1 shows an excerpt from the corpus.

O	SNR	Braxton is a gambling addict in deep to Mook (Ellen Burstyn), a local bookie.
S	SNR	Kennesaw is bitter about his marriage to a socialite (Rosanna Arquette), believing his wife to be unfaithful.
S	SR	The plot is twisty and complex, with lots of lengthy flashbacks, and plenty of surprises.
S	SR	However, there are times when it is needlessly complex, and at least one instance the storytelling turns so muddled that the answers to important plot points actually get lost.
s	SR	Take a look at L. A. Confidential, or the film's more likely inspiration, The Usual Suspects for how a complex plot can properly be handled.

Figure 6.1: Example passage from a document from the SR corpus with subjectivity (S/O) and sentiment relevance (SR/SNR) annotations

6.2.2 Sentiment Relevance vs. Subjectivity

We will next contrast sentiment relevance against two different notions of subjectivity. While there is a formal definition of subjectivity (cf. Section 3.1.2), in recent research, the term "subjectivity" has also been used to label approaches that are conceptually related but different from the traditional definition. Pang and Lee (2004) use a practical definition of subjectivity that suits their needs for training a machine learning system. They create their subjectivity corpus (henceforth P&L corpus, described in more detail in Appendix A.2.1) by gathering content from selected resources. As a rule, they extract subjective examples from quote snippets from Rotten Tomatoes,[14] a review aggregation website, each of which summarizes a review and is therefore highly subjective. Objective content is collected by sampling sentences from IMDb plot descriptions. These would in reality contain a mix of subjective and objective. We compare sentiment relevance to both the theoretical and empirical definition of subjectivity in the following experiments.

[14] http://www.rottentomatoes.com/

Annotation Experiment

First, we study agreement between human annotators, comparing manual sentiment relevance and subjectivity annotations. We had 762 sentences annotated for sentiment relevance by both annotators. We calculate inter-annotator agreement with Fleiss' κ and find an agreement of $\kappa = 0.69$. In addition, we obtained subjectivity annotations for the same data on Amazon Mechanical Turk, deciding each label through a vote of three, with an agreement of $\kappa = 0.61$. However, the agreement of the subjectivity and relevance annotations after voting, assuming that subjectivity equals relevance, is only at $\kappa = 0.48$. This suggests that there is indeed a measurable difference between subjectivity and relevance. We asked an independent annotator to examine the 225 examples where the annotations disagree who found that 83.5% of these cases are correctly identified as different.

The feedback we received from our annotators provided valuable insights into the difficulties of the sentiment relevance prediction problem. Both annotators found it difficult to annotate reviews of movies whose plot they were not familiar with. Thus, they often had to consult external sources like Wikipedia to read up on the movie before starting the annotation. They reported that they found the distinction between actor and character names particularly difficult and claimed that there are many ambiguous cases that could only be resolved with the knowledge of whether a person mentioned in the text was fictitious or not. This feedback in part inspired some of our later feature extraction methods. In addition, the annotators sometimes missed subtle expressions of sentiment, possibly due to insufficient second-language knowledge. This issue is to be expected when annotating in a second language, however in sentiment analysis, these issues are difficult to fix through the usual countermeasures such as improving annotation guidelines as the set of such expressions is too diverse.

Classification Experiment

We now show that although the P&L selection criteria (quote snippets vs. plot descriptions) bear resemblance to the definition of sentiment relevance, the two concepts are different. To show this, we assume that the subjective data is sentiment relevant and the objective data is sentiment nonrelevant. We divide both the

		test	
		P&L	**SR**
train	**P&L**	89.7	68.5
	SR	67.4	76.4

Table 6.1: TL/in-task $\bar{F}_1$ for P&L and SR corpora

vocabulary	$\mathbf{fp_{SR}}$	$\mathbf{fp_{SNR}}$
{actor, director, story}	0	7.5
{good, bad, great}	11.5	4.8

Table 6.2: Percentage of false positive sentences for sentiment relevant (fp_{SR}) and nonrelevant (fp_{SNR}) containing specific words. Training on P&L, evaluation on SR.

SR and P&L corpora into training and test set at a ratio of 50:50 and train the Stanford maximum entropy (MaxEnt) classifier with bag-of-word features.

Macro-averaged $\bar{F}_1$ for the four possible training-test combinations is shown in Table 6.1. The results indicate that the classes defined by the two labeled sets are different. A classifier trained on P&L performs worse by around 8% on SR than a classifier trained on SR (68.5 vs. 76.4). A classifier trained on SR performs worse by more than 20% on P&L than a classifier trained on P&L (67.4 vs. 89.7).

Note that the classes are not balanced in the sentiment relevance data while they are balanced in the subjectivity data. This can cause a misestimation of class probabilities and lead to the experienced performance drops. Indeed, if we either balance the sentiment relevance data or introduce an imbalance in the subjectivity data by undersampling in the P&L training set the minority class of the test set (SNR), we can significantly increase $\bar{F}_1$ to 74.8% and 77.9%, respectively, in the noisy label transfer setting. Note however that this step is difficult in practical applications if the actual label distribution is unknown. Also, in a realistic setting the distribution of the data is what it is – it cannot be adjusted to the training

set. We will conduct further experiments on balancing in Section 6.5 which show that an unsupervised sequence model is superior to artificial manipulation of class-imbalances.

Error analysis for the classifier trained on P&L shows that many sentences misclassified as sentiment relevant (fp_{SR}) contain polar words; e.g.,

(6.3) *Then, the situation turns bad.*

In contrast, sentences misclassified as sentiment nonrelevant (fp_{SNR}) contain named entities or plot and movie business vocabulary; e.g.,

(6.4) *Tim Roth delivers the most impressive acting job by getting the body language right.*

The word count statistics in Table 6.2 show this for three polar words and for three plot/movie business words. The P&L-trained classifier seems to have a strong bias to classify sentences with polar words as sentiment relevant even if they are not, perhaps because most training instances for the category *quote* are highly subjective, so that there is insufficient representation of less emphatic sentiment relevant sentences. These snippets rarely contain plot/movie-business words, so that the P&L-trained classifier assigns almost all sentences with such words to the category sentiment nonrelevant.

6.3 Methods

6.3.1 Discourse Constraints with Minimum Cut

It has been recognized that sentiment relevance, just like related phenomena such as subjectivity, has sequential properties (cf. Taboada et al., 2009). This means that sentences with the same property are likely to occur in sequence.

For this reason, it makes sense to impose the discourse constraint that a sentiment relevant (resp. sentiment nonrelevant) sentence tends to follow a sentiment relevant (resp. sentiment nonrelevant) sentence. This means that we need to optimize a trade-off between supervised and unsupervised information sources. This problem can be solved through graph segmentation using minimum cut (MinCut).

Following Pang and Lee (2004), we formalize the constraint by representing each document as a graph, encoding the strength of the constraint in the edges, and using MinCut to find the optimal solution.

For each document, consisting of N sentences, we create a discourse network, which is a directed graph $G = (\mathcal{V}, \mathcal{E}, \mathbf{W})$ with $N + 2$ nodes: N sentence nodes ($i \in \mathcal{V}$ for $i \in \{1, \ldots, N\}$, where i represents the document's i^{th} sentence) and a source ($s \in \mathcal{V}$) and a sink node ($t \in \mathcal{V}$). We define source and sink to represent the classes sentiment relevance and sentiment nonrelevance, respectively. Thus, we also refer to the source as `SR` and the sink as `SNR`.

We will now introduce two types of edges, *individual edges* and *association edges*. Individual edges connect sentence nodes to source or sink nodes. Each sentence node i has a directed edge $\langle s, i\rangle \in \mathcal{E}$ from the source to it and a directed edge $\langle i, t\rangle \in \mathcal{E}$ to the sink. The individual weights between a sentence i and the source/sink node ($\text{ind}(i, s) = w_{si}$ and $\text{ind}(i, t) = w_{it}$, respectively) are set according to some confidence measure for assigning it to the corresponding class. Association edges connect pairs of sentences. Each pair of sentence nodes i and j are connected through a directed edge $\langle i, j\rangle \in \mathcal{E}$ if $i < j$. Pang and Lee (2004) propose several ways of weighting association edges. We use squared decay, where the weight w_{ij} on the association edge is set to

$$w_{ij} = \frac{c}{(j - i)^2}.$$

c is a free parameter that controls the scaling of the association weights in relation to the individual weights. In practice, we did not find significant differences between the proposed measures. Pang and Lee (2004) also propose a threshold t on the sentence distance $(j - i) \leq t$ above which w_{ij} is set to zero. We found that this threshold does not affect our results significantly either, so we do not apply it (which can be formally expressed as $t = \infty$).

The resulting graph is a directed acyclic flow network from source to sink. On this graph, we compute the minimum cut. It solves the problem of trading-off the confidence of the classification decisions for "discourse coherence" Pang and Lee (2004). The discourse constraint has the effect that high-confidence labels are propagated over the sequence. As a result, outliers with low confidence are

eliminated and we get a "smoother" label sequence. To compute minimum cuts, we use the push-relabel maximum flow method (Cherkassky and Goldberg, 1995).

Unsupervised Parameter Optimization

We need to find values for multiple free parameters related to the sequence model. Unfortunately, we cannot perform supervised optimization on held-out data as we do not have any labeled data that includes sequence information. We therefore resort to a proxy measure. We first define a *run* as a sequence of sentences with the same label. The *run count* is the number of runs that occur within a document. For example, the excerpt shown in Figure 6.1 contains two runs, a sentiment nonrelevant and a sentiment relevant one. With this definition, we can compute the median and mean number of runs counts in the corpus, referring to them as the *median run count* and *mean run count*, respectively.

We set each parameter p to the value that produces a median run count for the predicted labels that is closest to the median run count in the corpus. In case of a tie, we also look at the proximity of the mean run count. We assume that the optimal median run count is known. In practice, it can be estimated from a small number of documents.[15] The median run count is non-differentiable. We do not we do not expect it to vary strongly under small changes of p, so we use grid search to find its optimal value.

6.3.2 Feature Extraction

Choosing features is crucial in situations where no high-quality training data is available, as it is the case in both distant supervision and transfer learning. We therefore introduce features which are more likely to support knowledge transfer. We are primarily interested in features that are robust and support generalization. We propose two linguistically motivated feature types for sentiment relevance classification that meet these requirements.

[15] Later experiments show that the estimate does not need to be too exact and can be "eyeballed".

Generalization through Semantic Features

To generalize over concepts, we make use of lexical taxonomies. A set of generalizations can be found by making a cut in the taxonomy hierarchy and defining the concepts there as base classes. For nouns, the taxonomy is WordNet (Miller, 1995) for which CoreLex (CX, Buitelaar, 1998) gives a set of basic types and classes. For verbs, we use VerbNet (VN, Kipper et al., 2008) which already contains base classes. These resources are described in more detail in Appendix A.2.2.

In order to generalize over verbs and nouns, it is necessary to have the corpus annotated with parts of speech. We use the automatic part-of-speech tagger by Bohnet (2010) to perform part-of-speech tagging as well as lemmatization (see Appendix A.1.3 for a description of the tool). The tagger returns Penn Treebank tags (Marcus et al., 1993), so nouns in the corpus will have a tag matching the pattern `NN.*`, and verbs will match `VB.*`.

We extend the existing feature representation by adding the base classes from the taxonomies as follows. For each noun, we look up the class in CoreLex and add it as a feature. For example, the noun *teacher* has the base class *human*, so we add *human* as a separate feature, which we write as `CX:human`. For each verb, we look up the base class in VerbNet and add it as a feature. For example, the verb *suggest* occurs in the VerbNet base class *say*, so we add a feature `VN:say` to the feature representation.

We refer to these feature sets as *CoreLex* (*CX*) and *VerbNet* (*VN*) features and to their combination as *semantic features* (*SEM*).

Named Entities

It has been recognized that in the movie domain, certain named entities correlate with either plot description or review content (Zhuang et al., 2006). Thus, using named entities as features in a sentiment relevance classifier is promising. However, as standard NER systems do not capture named entity types that are specialized to the movie domain, we cannot apply an out-of-the-box NER system. Instead, we opt for a lexicon-based approach similar to (Zhuang et al., 2006). We use the IMDb movie metadata database (Appendix A.2.3) from which we extract three different types of named entities:

- actors (<ACTOR>)
- directors, screenwriters, and composers (<PERSONNEL>)
- characters from movies (<CHARACTER>)

Many entries are unsuitable for NER, e.g., *dog* and *man* are frequently listed as a character. We address this problem by filtering out all entries in the database that also appear in lower case in a list of English words extracted from the dict.cc dictionary (Appendix A.2.3).

A name can be ambiguous between the types. For example, *John Williams* is a frequent name that occurs in all three categories. We disambiguate by calculating the maximum likelihood estimate of

$$p(t|n) = \frac{f(n,t)}{\sum_{t'} f(n,t')}$$

over all entries in the database, where n is a name, t is one of the three types, and $f(n,t)$ is the number of times n occurs in the database as type t. This estimate helps to disambiguate many examples for which a humanly salient choice exists. For our example, we learn that John Williams has the highest likelihood to be of the <PERSONNEL> type, corresponding with the association that the name John Williams in the movie business refers most likely to the composer. We also calculate these probabilities for all tokens that make up a name. While this may cause errors, it can also help in many cases where the name obviously belongs to a type. *Skywalker* (as in *Luke Skywalker*), for example, is very likely a character reference.

In addition, we use a set of simple co-reference rules to propagate annotations to related terms. If a capitalized word occurs, we check whether it is part of an already recognized named entity. For example, if we encounter *Robin* and we previously encountered *Robin Hood*, we assume that the two entities match. This rule has precedence over NER, so if a name matches a labeled entity, we do not attempt to label it through NER. Personal pronouns will match the most recently encountered named entity.

Data analysis reveals that apposition plays an important role in the structure of reviews. Many mentions of actors in parentheses are actually additional information for character mentions. As an example, in the sentence

(6.5) *Tracy Flick (Reese Witherspoon) is an over-achiever [...].*

the actor name *Reese Witherspoon* is only mentioned to say who the character was played by. Thus, we always interpret a name preceding an actor in parentheses as a character mention. This serves two purposes. First, we avoid adding unhelpful `<ACTOR>` features. Second, we can recognize character mentions for which IMDb provides insufficient information. This feature is written as `<ACTOR_P>`. The set of all features described above is referred to as *named entities* (*NE*).

Sequential Features

Following previous sequence classification work with maximum entropy models (e.g., Ratnaparkhi, 1996), we use selected features of adjacent sentences. If a sentence contains a feature `F`, we add the feature to the following sentence, indicating that it was inherited from the previous sentence. We write this new feature as `F+1`. For example, if a `<CHARACTER>` feature occurs in a sentence, the feature `<CHARACTER+1>` is added to the following sentence. For sentiment relevance classification, we perform this operation only for NE features as they are restricted to a few classes and thus will not enlarge the feature space notably. We refer to this feature set as *sequential features* (*SQ*).

6.4 Distant Supervision

Since a large labeled resource for sentiment relevance classification is not yet available, we investigate semi-supervised methods for creating sentiment relevance classifiers. In this section, we show how to bootstrap a sentiment relevance classifier by distant supervision (DS) .

Even though we do not have sentiment relevance annotations, there are sources of metadata for the movie domain that we can leverage for distant supervision. Specifically, movie databases like IMDb contain both metadata about the plot, in particular the characters of a movie, and metadata about the "creators" who were involved in the production of the movie: actors, writers, directors, and composers. We will now make the following assumptions: Statements about characters usually describe the plot and are not sentiment relevant. Statements about the

creators tend to be evaluations of their contributions – positive or negative – to the movie. Based on these assumptions, we formulate a classification rule: Count occurrences of NE features and label sentences that contain a majority of creators, i.e. `<ACTOR>` and `<PERSONNEL>` features, as `SR` and sentences that contain a majority of `<CHARACTER>` features as `SNR`. Ties are labeled with the majority class, `SR`. This is a form of distant supervision in that we use the IMDb metadata database as described in Section 6.3.2 to automatically label sentences based on which feature from the database they contain.

6.4.1 Initial Distant Supervision Experiment

The simple labeling rule described above covers 1583 sentences with an $\bar{F}_1$ score of 67.2%. We call these labels inferred from NE metadata *distant supervision (DS) labels*. To increase coverage, we train a MaxEnt classifier (Stanford classifier, cf. Appendix A.1.1) on the labeled examples. This model achieves an $\bar{F}_1$ of 61.2% on the SR corpus (Table 6.3, line 2). As this classifier uses training data that is biased towards a specialized case (sentences containing the named entity types actors and characters), it does not generalize well to other sentiment relevance problems and thus yields lower performance on the full dataset. This distant supervision setup suffers from two issues. First, the classifier only sees a subset of examples that contain named entities, making generalization to other types of expressions difficult. Second, there is no way to control the quality of the input to the classifier, as we have no confidence measure for our distant supervision labeling rule.

We will address these two issues by introducing an intermediate step, the unsupervised sequence model introduced in Section 6.3.1. We thus first apply MinCut as described in the following paragraphs and then select the most confident examples as training material to bootstrap the supervised classifier.

6.4.2 Further Experimental Setup

MinCut Setup

We follow the general MinCut setup described in Section 6.3.1. Each document is represented as a graph of sentence nodes and special source/sink nodes which

represent SR/SNR. As explained above, we next assume that actors and directors indicate relevance and characters indicate nonrelevance. Accordingly, we define $f_{\texttt{SR}}$ to be the number of <ACTOR> and <PERSONNEL> features occurring in a sentence, and $f_{\texttt{SNR}}$ the frequency of <CHARACTER> features. We then set the individual weight between a sentence node i and the source/sink nodes to $\text{ind}(i, x) = f_x$ where $x \in \{\texttt{SR}, \texttt{SNR}\}$.

The MinCut parameter c is set to 1; we wish to give the association scores high weights as (i) the individual scores are unreliable due to sparseness of the named entity features, and (ii) there might be long spans that have individual weights with zero values. Figure 6.2 shows an example graph created with this method. We will discuss this example in more detail during error analysis.

Confidence-Based Data Selection

We use the output of the base classifier to train supervised models. Since the MinCut model is based on a weak assumption, it will make many false decisions. To eliminate them, we use only those documents as training data where the base classifier has confidence. As the confidence measure for a document, we use the maximum flow value v – the maximal "amount of fluid" flowing through the document. The max-flow min-cut theorem (Ford and Fulkerson, 1956) states that if the flow value is low, then the cut was found by cutting edges with a lower weight sum. Thus, we assume that it was easier to calculate. This means that the document is more likely to have been segmented correctly. Following this assumption, we train a MaxEnt classifier on the k% of documents that have the lowest maximum flow values v, where k is a parameter which we optimize using the run count method introduced in Section 6.3.1.[16]

6.4.3 Experiments and Results

Table 6.3 shows sentiment relevant ($F_1^{(\texttt{SR})}$), sentiment nonrelevant ($F_1^{(\texttt{SNR})}$) and macro average ($\bar{F}_1$) F_1 values for different setups with this parameter. We compare the following basic setups:

[16] Grid search on $[0, 100]$ with a step size of 10 yields an optimum for $k = 40\%$ corresponding to a maximum flow value $v \leq 4000$.

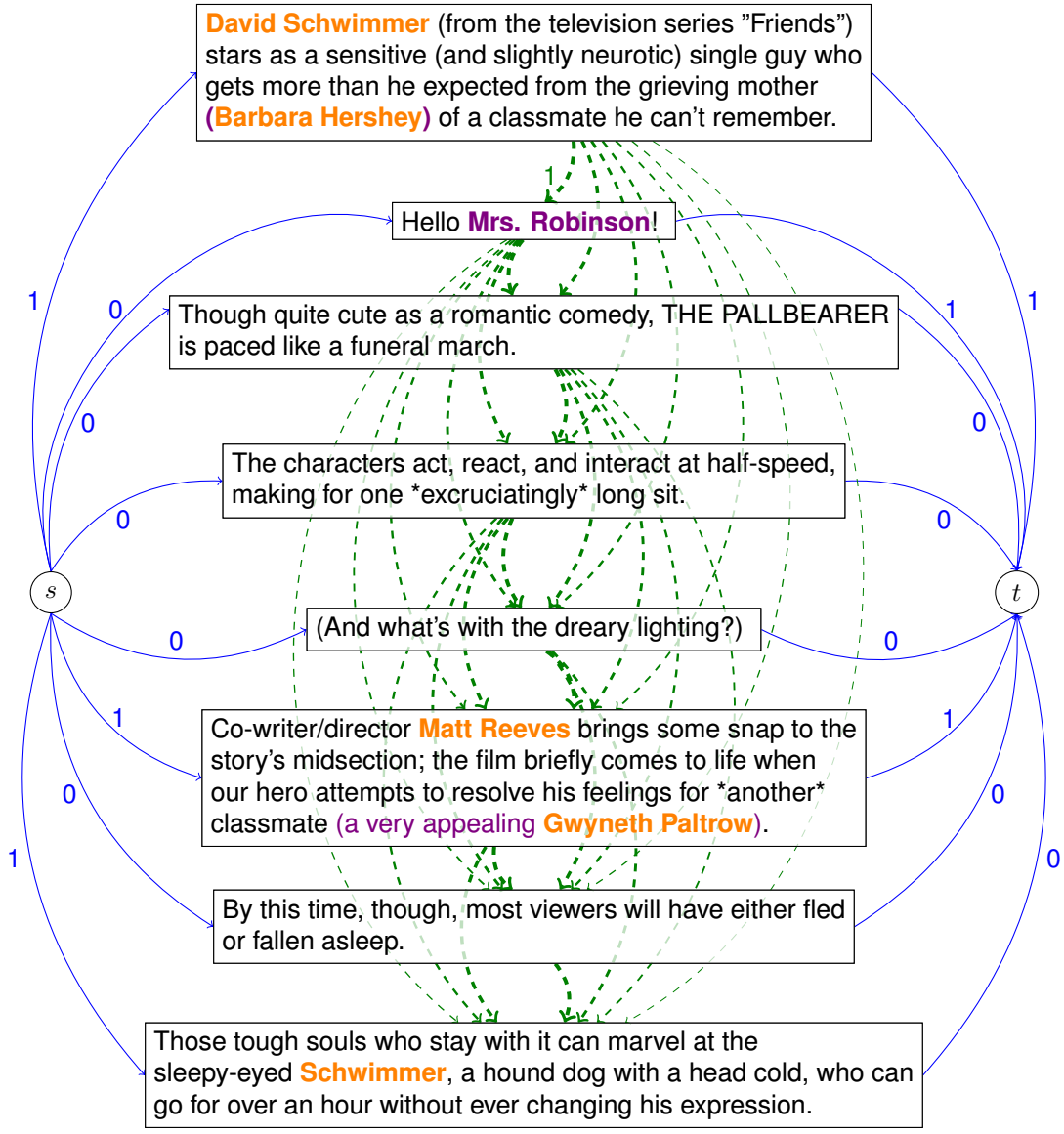


Figure 6.2: Example distant supervision graph for the negative document #671 from the Pang & Lee movie corpus showing edge weights from the IMDb database. Individual edges are shown in blue, association edges in green. Character NE's are shown in purple, personnel NE's in orange. Phrases in parentheses containing an NE type change the label and are marked accordingly.

	Model	Features	$F_1^{(\texttt{SR})}$	$F_1^{(\texttt{SNR})}$	$\bar{F}_1$
1	Majority BL	–	88.3	0.0	44.2
2	MaxEnt (DSlabels)	NE	79.8	42.6	61.2^{1}
3	DSMinCut	NE	79.6	48.2	63.9^{12}
4	DSMinCut + MaxEnt	NE	84.8	46.4	65.6^{12}
5	DSMinCut + MaxEnt	NE+SEM	85.2	48.0	66.6^{124}
6	DSMinCut + MaxEnt	NE+SQ	84.8	49.2	67.0^{1234}
7	DSMinCut + MaxEnt	NE+SQ+SEM	84.5	49.1	66.8^{1234}

Table 6.3: Distant supervision classification results: $F_1^{(\texttt{SR})}$, $F_1^{(\texttt{SNR})}$ and $\bar{F}_1$. Superscript numbers indicate a significant improvement over the corresponding line.

1. The majority baseline (BL) when choosing the most frequent label (SR).
2. A MaxEnt baseline trained on DS labels without application of MinCut.
3. The base classifier using MinCut as described above (referred to as DSMinCut).

Further, we train supervised MaxEnt classifiers (lines 4–7) based on the labels from DSMinCut (line 3), using the named entity (NE), semantic (SEM), and sequential (SEQ) features introduced in Section 6.3.2.

We test statistical significance using the approximate randomization test with document permutations at $p < .05$. We achieve classification results above baseline using the DSMinCut base classifier (line 3) and a considerable improvement through distant supervision. We found that all classifiers using DS labels and Mincut are significantly better than MaxEnt trained on purely rule-based DS labels (line 2). Also, the MaxEnt models using SQ features (lines 6 and 7) are significantly better than the MinCut base classifier (line 3). For comparison to a chain-based sequence model, we train a 2^{nd} order Conditional Random Field classifier (CRF, McCallum, 2002). However, the improvements over the corresponding MaxEnt models are small ($\leq 1\%$) and not statistically significant.

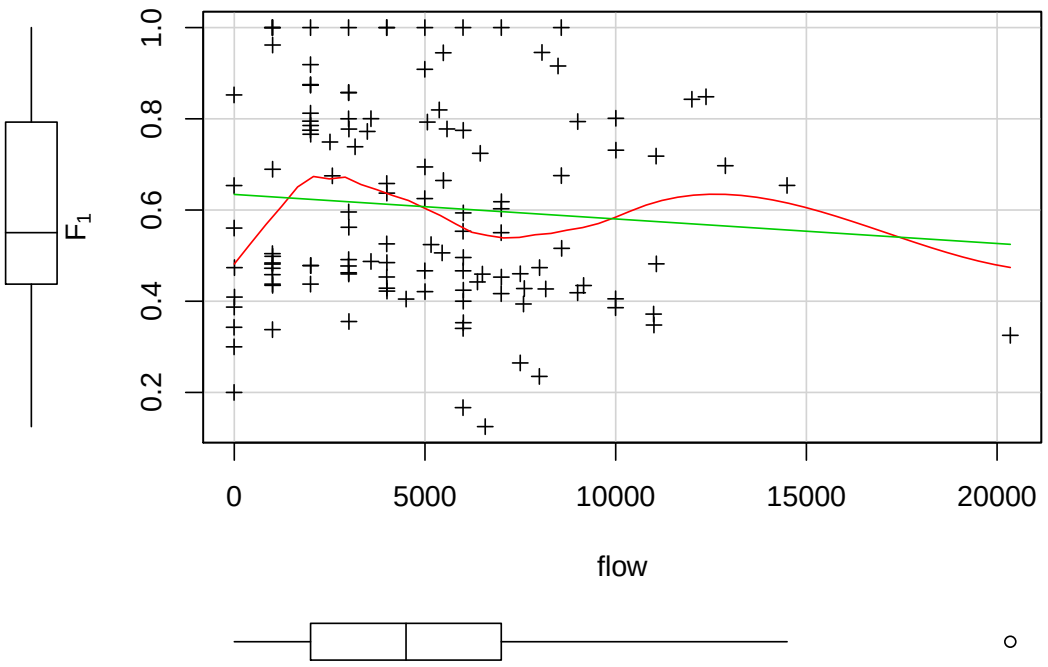


Figure 6.3: Base classifier results by document: $\bar{F}_1$ measure vs. flow. Green line: linear regression, red line: locally weighted regression.

We found that both semantic (lines 5,7) and sequential (lines 6,7) features help to improve the classifier. The best model (line 6) performs better than MinCut (3) by 3.1% and better than training on purely rule-generated DS labels (line 2) by 5.8%. However, we did not find a cumulative effect (line 7) of the two feature sets. We will examine minimum cut and the semantic features in more detail next.

Minimum Cut

To verify that our parameter optimization method selected a reasonable flow value threshold, we investigate the correlation of $\bar{F}_1$ and flow. Figure 6.3 shows $\bar{F}_1$ and the flow value for each document as a point in the plot. Fitting a linear regression

model (green line) to the data reveals that lower flow values are correlated with slightly higher $\bar{F}_1$. As the relation between the variables does not appear to be strictly linear, we apply locally weighted regression (red curve, Cleveland and Devlin, 1988). It shows two peaks around flow values of 2500 and 12000. This suggests that the maximum flow value threshold of 4000 is a reasonable choice – $\bar{F}_1$ tends to decrease rapidly after this point and only recovers much later.

Looking at the sentence graphs in more detail, we can identify several reasons for why the model fails in some cases. Take the graph shown in Figure 6.2 as an example. First, ambiguous examples are difficult to handle, such as the 1^{st} 6^{th} sentences, which contain both sentiment relevant and nonrelevant content. One possibility to address this issue would be a better segmentation of the data – possibly supported by an sentiment relevance classifier – at the sub-sentence level. Alternatives may include machine learning extensions such as hierarchical models. A second problem is that in cases like the example, there are large gaps between nodes with correct seed information. In the example, only the first, second, and last node actually contribute a non-zero individual score, which makes the prediction of the correct labels for nodes in the middle more difficult. This problem arises due to our choice of named entities as the source of distant supervision, which, while being helpful indicators, occur only sparsely in reviews. Third, the distant supervision rules are noisy, as it is evident in the 1^{st} sentence which contains an actor mention despite being sentiment nonrelevant.

Semantic Features

We will next analyze the structure of the bootstrapped classifiers. Examining the highest-weighted features of the best model using DS labels (line 6) shows that NE features are dominant (Table 6.4). This correlation is not surprising as the seed labels were induced based on NE features. Thus, it may as well be a sign of overfitting. The top weighted feature shown in the table is the actor apposition feature `<ACTOR_P>` resulting from our parentheses rule. As expected, this feature and its sequential counterpart `<ACTOR_P+1>` is a good indicator for non-relevant content.

Words like *see* and *theater* that are commonly used to describe a movie theater

feature	w_{SR}
<ACTOR_P>	1.83
<CHARACTER>	1.27
<ACTOR>	-1.15
<CHARACTER+1>	1.01
<ACTOR+1>	-0.86
<ACTOR_P+1>	0.85
<PERSONNEL+1>	-0.81
see	0.80
remember	0.78
theater	0.72
bad	0.70
well	-0.70
kilgore	0.61
previews	0.60
tame	0.57
debate	0.54
delightful	0.53
dangerous	0.53
thrown	0.53
things	0.53
<PERSONNEL>	0.42

Table 6.4: 20 highest weighted features and NE & SQ features in the best-performing model. w_{SR} is the weight in the sentiment relevant class.

visit are considered highly relevant. Also relevant are subjective words, e.g., *bad*, *tame*, and *delightful*. Interestingly, some subjective features have high weights for sentiment nonrelevance, such as *horrible* with $w_{\mathrm{SNR}} = 0.32$. These words could be thematically fitting for non-relevant content, however this could also indicate that the classifier is overfitting to the training data.

Generally, the quality of NER is crucial in this task. While IMDb is in general a thoroughly compiled database, it is not perfect. For example, all main characters in *Groundhog Day* are listed with their first name only even though the full names are given in the movie, which poses a difficulty for our pattern matching NER approach. Some entries are even left incomplete intentionally to avoid spoiling the plot. The data also contains ambiguities between characters and titles (e.g., *Forrest Gump*) that are impossible to resolve with our maximum likelihood method. In some types of movies, e.g., documentaries, the distinction between characters and actors makes little sense. Furthermore, ambiguities like occurrences of common names such as *John* are impossible to resolve if there is no earlier full referring expression.

6.4.4 Conclusion

The results of our experiments using distant supervision show that a sentiment relevance classifier can be trained successfully by labeling data with a couple of simple feature rules, with MinCut-based input significantly outperforming the baseline. Named entity recognition, accomplished with data extracted from a domain-specific database, plays a significant rule in creating an initial labeling.

6.5 Transfer Learning

One reason for the low performance of the distant supervision approach is that it uses only a small amount of training data. Less than 50% of the corpus are selected for training. To address this problem, we now investigate a second semi-supervised method for sentiment relevance classification, transfer learning (TL). In our transfer learning setting, we will train a classifier on a related but different task, and use it for sentiment relevance prediction. This way, we can exploit external

labeled data which might be available in significantly larger quantities.

In transfer learning, the key to success is to find a generalized feature representation that supports knowledge transfer. We use our semantic feature generalization method to introduce such features. We again use MinCut to impose discourse constraints. This time, we first classify the data using a supervised classifier and then use MinCut to smooth the sequences.

6.5.1 Experimental Setup

Data, Feature Extraction, and Classification Setup

For training our supervised base classifier, we use the P&L movie subjectivity corpus (Pang and Lee, 2004, cf. Appendix A.2.1). For evaluation, we use the SR corpus.

The P&L corpus consists of 5000 highly subjective *quote* review snippets from Rotten Tomatoes and 5000 *plot* sentences from IMDb plot descriptions which the authors use as objective data. As the data was collected automatically using heuristics, the resulting labels can be either viewed as noisy sentiment relevance labels or noisy subjectivity labels. Compared to distant supervision on a database, the advantage of training on P&L is that the training set is much larger, containing around 7 times as much data as compared to the amount selected for supervised training in the distant supervision experiment.

We assume that the *quote* examples are sentiment relevant and the *plot* examples are sentiment nonrelevant, which constitutes a transfer learning setup from subjectivity to sentiment relevance.

A drawback of the published version of the P&L corpus is that capitalization was removed, which makes the extraction of NE features difficult. For this reason, we extract only the semantic features (SEM).

The baseline (BL) uses a simple bag-of-words representation of sentences for classification which we then extend with semantic features. We train a MaxEnt classifier for each of the feature sets.

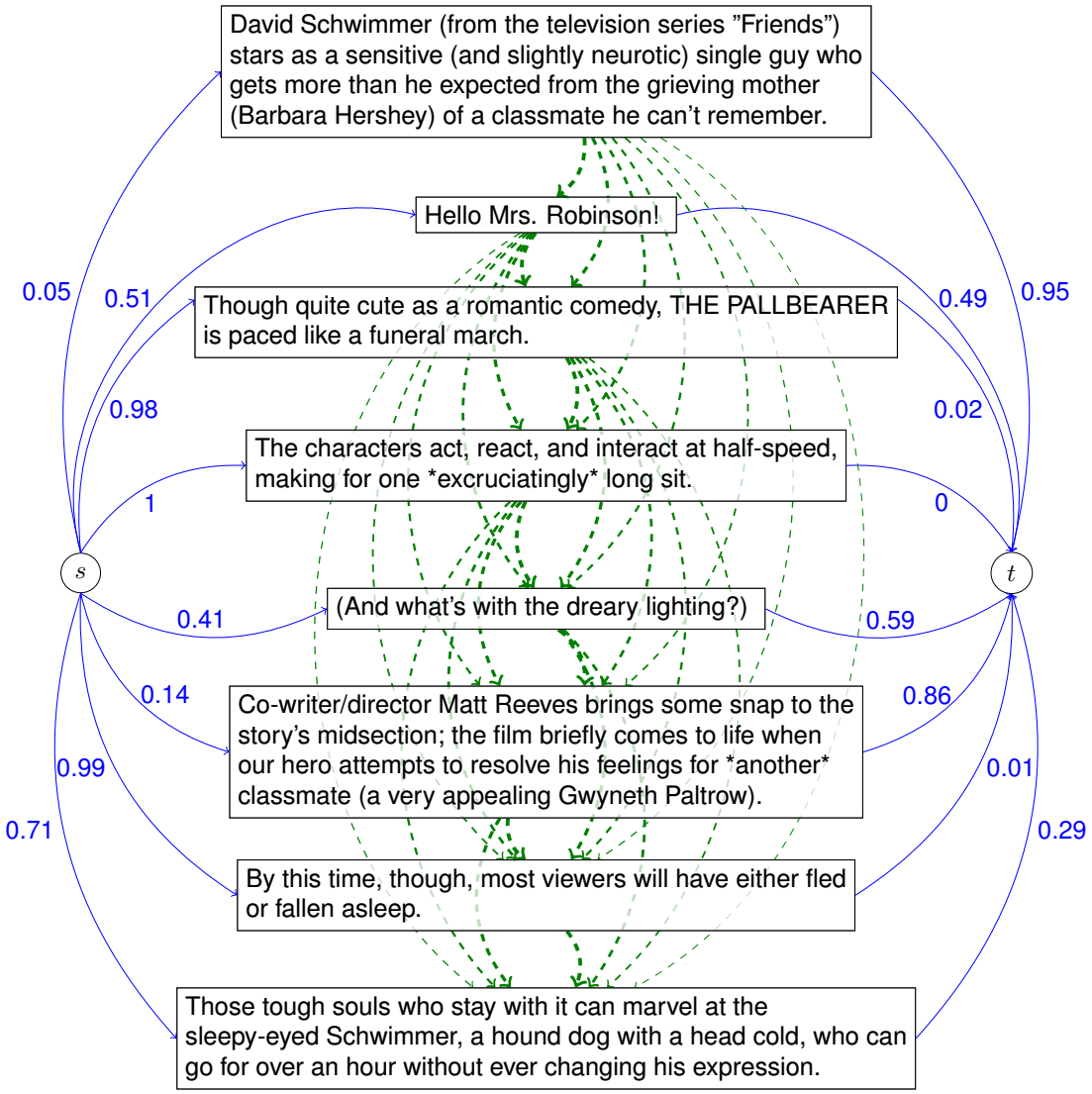


Figure 6.4: Example transfer learning graph for the negative document #671 from the Pang & Lee movie corpus showing edge weights from the best-performing model. Individual edges are shown in blue, association edges in green.

Model	base classifier			MinCut		
	$F_1^{(\texttt{SR})}$	$F_1^{(\texttt{SNR})}$	$\bar{F}_1$	$F_1^{(\texttt{SR})}$	$F_1^{(\texttt{SNR})}$	$\bar{F}_1$
1 BL	81.1	58.6	69.9	87.2	67.8	77.5^{B}
2 CX	82.9	60.1	71.5^{B}	89.0	70.3	79.7^{BM}
3 VN	85.6	62.1	73.9^{B}	91.4	73.6	82.5^{BM}
4 CX+VN	88.3	62.9	75.6^{B}	92.7	72.2	82.5^{BM}

Table 6.5: Transfer learning classification results: $F_1^{(\texttt{SR})}$, $F_1^{(\texttt{SNR})}$ and $\bar{F}_1$. B indicates a significant improvement over the BL base classifier, M over BL MinCut.

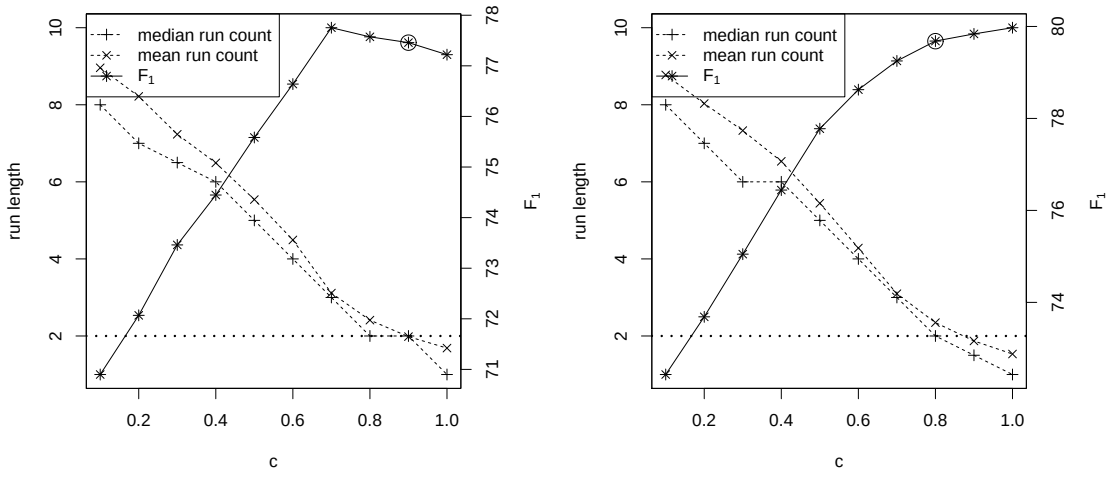
MinCut Setup

We again implement the basic MinCut setup from Section 6.3, constructing a graph for each document. We set the individual weight $\text{ind}(i, x)$ on the edge between sentence node i and source or sink node $x \in \{\texttt{SR}, \texttt{SNR}\}$ to the estimate $p(x|i)$ returned by the MaxEnt base classifier. Figure 6.4 shows an example of a graph constructed in this setup. The parameter c of the MinCut model is tuned using the run count method described in Section 6.3.1.[17]

6.5.2 Experiments and Results

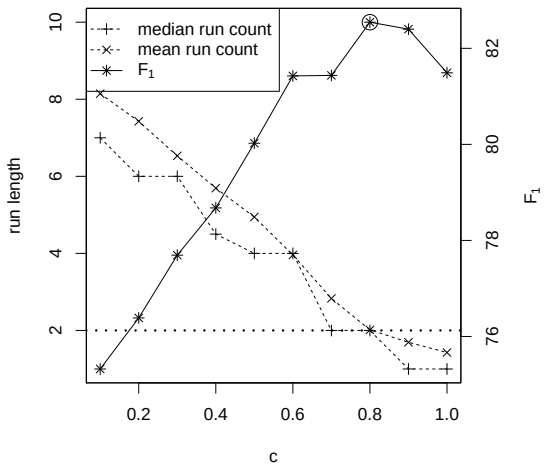
Table 6.5 shows the performance of the MaxEnt base classifiers and MinCut classifiers for different feature sets. As we would expect, the baseline performance of the supervised classifier evaluated on the SR corpus is low: 69.9% (Table 6.5, line 1). MinCut significantly boosts the performance by 7.9% to 77.5% (line 1), a result similar to (Pang and Lee, 2004). Adding semantic features improves supervised classification significantly by 5.7% (75.6% on line 4). When MinCut and both types of semantic features are used together, these improvements are partially cumulative: an improvement over the baseline by 12.6% to 82.5% (line 4). In the following sections, we investigate the influence of both aspects in more detail.

[17] Grid search on $[0.1, 1]$ yields values of $c \in \{0.8, 0.9\}$, depending on the experiment (cf. Figure 6.5).

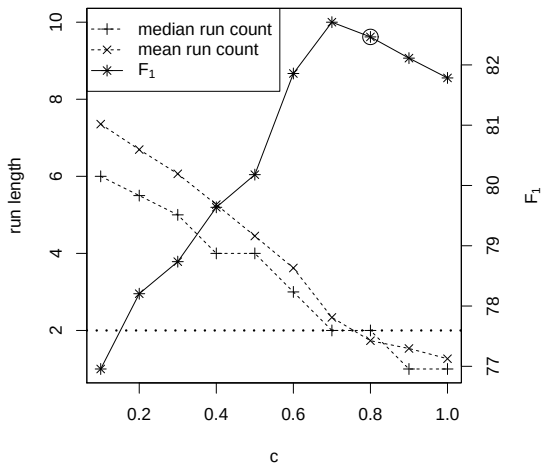


(a) BL

(b) CX



(c) VN



(d) CX+VN

Figure 6.5: $\bar{F}_1$ measure for different values of c. Horizontal line: optimal median run count. Circle: selected point.

Minimum Cut

To illustrate the run-based parameter optimization criterion, we show $\bar{F}_1$ and median/mean run lengths for different values of c for the four setups from Table 6.5 in Figure 6.5. Due to differences in the base classifier, the optimum of c varies between the experiments. The optimization criterion does not always correlate perfectly with $\bar{F}_1$. However, we found no statistically significant difference between the selected result and the highest $\bar{F}_1$ value.

We also experiment with a training set where an artificial class imbalance is introduced, matching the 80:20 imbalance of `SR`:`SNR` in the sentiment relevance corpus. The result of this experiment is shown in Table 6.6. The BL and CX base classifiers perform significantly better than their balanced counterparts (cf. Table 6.5). However, with the introduction of VN, results drop markedly. This may be attributed to a bias towards the majority class that is amplified when using the VN feature set. After applying MinCut, we found that while the results for BL with and without imbalances does not differ significantly. However, models using CX and VN features and imbalances are actually significantly inferior to the respective balanced versions. While the artificially imbalanced models have the potential to perform better in the base classifier setup, the balanced models seem to work better in combination with MinCut, producing the best overall model. This result suggests that MinCut is more effective at coping with class imbalances than artificial adjustments to the class distribution.

This experiment shows why MinCut is successful for transfer learning. It can exploit test set information without supervision as the MinCut graph is built directly on each test set review. If high-confidence information is "seeded" within a document and then spread to neighbors, mistakes with low confidence are corrected. This way, MinCut also leads to a compensation of differences in class distribution.

Once again, we conduct some error analysis of the graphs produced by the model. Looking at Figure 6.4, we see that as expected, not all classification decisions made by the base classifier were correct, such as 5^{th} sentence (*... dreary lighting ...*) which received a `SNR` label. However, since the sentence occurs in the neighborhood of nodes that are mostly predicted to be `SR` with high confidence, this mistake can be corrected through MinCut. In turn, the MinCut model may be misled by

Model	base classifier			MinCut		
	$F_1^{(\mathtt{SR})}$	$F_1^{(\mathtt{SNR})}$	$\bar{F}_1$	$F_1^{(\mathtt{SR})}$	$F_1^{(\mathtt{SNR})}$	$\bar{F}_1$
1 BL	89.3	60.4	74.8†	92.6	65.9	79.3
2 CX	90.0	62.2	76.1†	92.4	64.2	78.3
3 VN	90.3	57.9	74.1	91.6	54.6	73.1†
4 CX+VN	90.3	58.6	74.5	90.3	58.6	74.5†

Table 6.6: Transfer learning results after introducing an imbalance to the training set: $F_1^{(\mathtt{SR})}$, $F_1^{(\mathtt{SNR})}$ and $\bar{F}_1$. † indicates a significant difference to the respective cell in Table 6.5.

confident decisions for ambiguous nodes like the one representing the 6$^{\text{th}}$ sentence.

Semantic Features

The results presented above are evidence that semantic features are robust to the differences between the classification tasks subjectivity and sentiment relevance that we documented in Table 6.1. We will now take a look at the highest-weighted features of the best-performing model (Table 6.7).

Meaningful semantic feature classes receive high weights. We will briefly review some of them. A table explaining the relevant CoreLex class abbreviations can be found in Appendix A.2.2. As an example, for nouns, the *human groups* class (*CX:hum*) from CoreLex which among other entries contains professions that are frequently associated with non-relevant plot descriptions. *CX:gsl* contains nationality nouns. High-weighted VerbNet base classes include *VN:murder* and *VN:cope*, containing verbs relevant to plot descriptions (e.g., *kill* and *manage*) which are highly nonrelevant. The base class *VN:light_emission* contains verbs such as *shine* which also have a subjective meaning, which is why the classifier selects it as a good feature for sentiment relevance. This result may be a sign of the metaphorical nature of sentiment (e.g., bright is good, dark is bad, as described by Lakoff and Johnson, 1980). Interestingly, pronouns are recognized as strong features. *I* is a good feature for sentiment relevant content as self-reference indicates that the

feature	w_{SR}
–	1.56
its	1.17
director	1.01
he	-1.00
book	-0.99
material	0.99
interesting	0.98
screen	0.98
she	-0.97
here	0.96
my	0.94
performance	0.93
they	-0.93
him	-0.92
-	-0.91
evil	-0.90
film's	0.88
i	0.88
characters	0.87
kung-fu	-0.86

feature	w_{SR}
CX:agh	-0.70
CX:pas	-0.70
VN:tape	0.66
VN:occurrence	-0.65
VN:cope	-0.60
CX:gsl	-0.59
CX:lor	-0.57
VN:battle	-0.56
CX:ara	0.54
VN:coil	0.54
VN:light_emission	0.51
VN:begin	-0.48
CX:atp	0.47
CX:hum	-0.47
VN:turn	-0.46
VN:engender	-0.45
CX:atr	0.45
VN:appeal	0.44
VN:establish	-0.44
VN:murder	-0.43

Table 6.7: 20 highest-weighted features in the best-performing model. Left: all features, right: CX and VN features. w_{SR} is the weight in the sentiment relevant class.

reviewer is stating their opinion. Third-person pronouns (*he*, *she*, and *they*) occur more frequently in sentiment non-relevant examples, mostly in long passages with plot descriptions.

6.5.3 Conclusion

These experiments demonstrate that sentiment relevance classification improves considerably through transfer learning if semantic feature generalization and unsupervised sequence classification through MinCut are applied. We found that our unsupervised selection criterion performs well and provided further analysis of the relation of MinCut to class distribution balancing. We showed that our feature sets provide useful generalization over semantic concepts.

6.6 Related Work

6.6.1 Related Concepts

Conceptually, our work is most closely related to (Taboada et al., 2009) who define a fine-grained classification that is similar to sentiment relevance on the highest level. However, unlike our study, they do not experimentally compare their classification scheme to prior work in their experiments to show that this scheme is different. In addition, they work on the paragraph level. However, paragraphs often contain a mix of sentiment relevant and sentiment nonrelevant sentences. We use the minimum cut method and are therefore able to incorporate discourse-level constraints in a more flexible fashion, giving preference to "relevance-uniform" paragraphs without mandating them.

Täckström and McDonald (2011) develop a fine-grained annotation scheme that includes sentiment nonrelevance as one of five categories. However, they do not use the category sentiment nonrelevance directly in their experiments and do not evaluate classification accuracy for it. We do not use their dataset as we target the movie domain for the aforementioned quality reasons. Changing both the domain (movies to products) and the task (subjectivity to sentiment relevance) would give rise to interactions that we wanted to avoid in our study.

The notion of annotator rationales (Zaidan et al., 2007) has some overlap with our notion of sentiment relevance. Yessenalina et al. (2010) use rationales in a multi-level model to integrate sentence-level information into a document classifier. Neither paper presents a direct gold standard evaluation of the accuracy of rationale detection.

Sentiment relevance is also related to review mining (e.g., Ding et al., 2008) and similar sentiment retrieval techniques (e.g., Eguchi and Lavrenko, 2006) in that they aim to find phrases, sentences or snippets that are sentiment relevant for sentiment, either with respect to certain features or with a focus on high-precision retrieval (cf. Liu, 2010). However, finding a few sentiment relevant items with high precision is much easier than the task we address: exhaustive classification of all sentences.

In the past, subjectivity has been used to detect nonrelevant content, so we view it as a related concept as well. Many publications have addressed subjectivity in sentiment analysis. Two important papers that are based on the original philosophical definition of the term (internal state of mind vs. external reality) are (Wilson and Wiebe, 2003) and (Riloff and Wiebe, 2003). As we argue above, if the goal is to identify parts of a document that are useful/non-useful for sentiment analysis, then sentiment relevance is a better notion to use.

Researchers have implicitly deviated from the philosophical definition because they were primarily interested in satisfying the needs of a particular task. For example, Pang and Lee (2004) use a minimum cut graph model for review summarization. Because they do not directly evaluate the results of subjectivity classification, it is not clear to what extent their method is able to identify subjectivity correctly.

6.6.2 Distant Supervision and Transfer Learning

Täckström and McDonald (2011) solve a similar sequence problem by applying a distantly supervised classifier with an unsupervised hidden sequence component. Their setup differs from ours as our focus lies on pattern-based distant supervision instead of distant supervision using documents for sentence classification. They also work on a different task, polarity prediction. Transfer learning has been ap-

plied previously in sentiment analysis (Tan and Cheng, 2009), targeting polarity detection.

6.6.3 Feature Extraction

Named-entity features in movie reviews were first used by Zhuang et al. (2006), in the form of feature-opinion pairs (e.g., a positive opinion about the acting). They show that recognizing plot elements (e.g., script) and classes of people (e.g., actor) benefits review summarization. We follow their approach by using IMDb to define named entity features. We extend their work by introducing methods for labeling partial uses of names and pronominal references and taking into account the maximum likelihood estimate for disambiguation. We address a different problem (sentiment relevance vs. opinions) and use different methods (graph-based and statistical vs. rule-based).

6.6.4 Conclusion

In summary, no direct evaluation of sentiment relevance has been performed previously. One of our contributions is that we provide a single-domain gold standard for sentiment relevance, created based on clear annotation guidelines, and use it for direct evaluation.

Another contribution is that we show that generalization based on semantic classes improves sentiment relevance classification. While previous work has shown the utility of other types of feature generalization for sentiment and subjectivity analysis (e.g., syntax and part-of-speech, Riloff and Wiebe, 2003), semantic classes have so far not been exploited.

6.7 Summary

A number of different notions, including subjectivity, have been proposed for distinguishing parts of documents that convey sentiment from those that do not. We introduced *sentiment relevance* to make this distinction and argued that it better reflects the requirements of sentiment analysis systems. Our experiments

demonstrated that sentiment relevance and subjectivity are related but different.

Since a large labeled sentiment relevance resource does not yet exist, we investigate semi-supervised approaches to sentiment relevance classification that do not require sentiment relevance labeled data. We show that a combination of different techniques gives us the best results: semantic generalization features, imposing discourse constraints implemented as the minimum cut graph-theoretic method, automatic "distant" labeling based on a domain-specific metadata database and transfer learning to exploit existing labels for a related classification problem.

There are several possible directions for future work on the sentiment relevance classification problem. First, the obvious combination of transfer learning and distant supervision has not been explored yet. Using both approaches jointly could potentially yield further improvements. In addition, distant supervision from the paragraph-labeled data by Taboada et al. (2009) is possible. Another research direction is domain adaptation. As mentioned above, a similar dataset exists for different product review domains. Using further feature-space modifying techniques that have been promising in polarity classification (e.g., Glorot et al., 2011), domain adaptation could be attempted for sentiment relevance.

7 Compositionality of Recursive Autoencoders

7.1 Introduction

In the previous chapters, we have addressed sentiment analysis exclusively with the clue model. We have experienced both upsides and downsides of the model. It works well even when faced with a small set of indicators, however, it becomes difficult to improve beyond that point. Most crucially, the model is very sensitive to domain (cf. Chapter 5) and topic effects (cf. Chapter 6). Further, sentiment may in general be expressed through statements of arbitrary complexity. Interestingly, we often find a clue at the core of such statements which is then modified throughout the structure. This has been recognized previously, for example by Polanyi and Zaenen (2006) who formalize these mechanisms of modification as "polarity shifters" (cf. Section 3.2).

There have been efforts to automatically learn polarity shifters (e.g., Choi and Cardie, 2008; Wiegand et al., 2010), however, most of them focus on specific syntactic constructions and thus have very limited coverage. Recent research on fully compositional models for sentiment analysis (e.g., Yessenalina and Cardie, 2011) promises holistic approach that addresses these limitations. In this chapter, we will provide a more in-depth analysis of a compositional model that makes use of neural networks.

Deep neural networks (DNNs) have been gaining increasing attention in Natural Language Processing. Many of the traditional NLP tasks have been addressed, such as syntactic parsing (Socher et al., 2010), semantic role labeling (Collobert et al., 2011), machine translation (Deselaers et al., 2009), and document classification (Glorot et al., 2011).

A frequently cited DNN model for NLP tasks was introduced by (Socher et al., 2011), the *Semi-Supervised Recursive Autoencoder* (RAE). This model pursues a pairwise word composition strategy, representing each word as a vector and

recursively applying an autoencoder function to unify pairs of nodes, yielding a binary tree. One appealing property of RAEs is that the resulting structure lends itself to a syntactic or even semantic interpretation. However, so far, no in-depth analysis of RAEs in terms of these structures has been performed.

Our main interest in this chapter is to analyze the behavior of RAEs as a compositional model. We will investigate the following key questions: (i) Are the structures produced by RAEs interpretable syntactically or semantically? (ii) Can the structures be simplified? Addressing these questions will show whether the RAE is compositional both from a linguistic and a machine learning point of view.

We will analyze these points empirically in a sentiment classification task for which the RAE has been applied previously (Socher et al., 2011). We introduce two methods for analysis. First, we let humans rate the structures according to syntactic and semantic criteria. Second, we try to simplify the RAE structures automatically and evaluate the resulting models on a classification task.

The rest of the chapter is structured as follows. In Section 7.2, we describe RAEs, particularly highlighting some details regarding implementation. We then introduce different ways of structural simplification in Section 7.3. We present two types of experiments in Section 7.4. Section 7.4.2 contains error analysis of RAEs conducted by human annotators. In Section 7.4.3 we carry out the experiments on structural simplifications.

7.2 Semi-Supervised Recursive Autoencoders

The central model in this chapter is the Semi-Supervised Recursive Autoencoder (RAE, Socher et al., 2011). This section describes this model and discusses some important implementation details.[18]

The RAE is a structural model for sequences. It recursively applies an autoencoder to construct a tree structure over the words in a sentence. Each word, i.e., each leaf of the tree, is represented by vector which is independent of the context in which the word occurs. Each non-leaf node represents a feature vector which is the encoding yielded by an autoencoder of the feature vectors of its children.

[18]Some of these details can be found in the RAE implementation available at `http://www.socher.org/`.

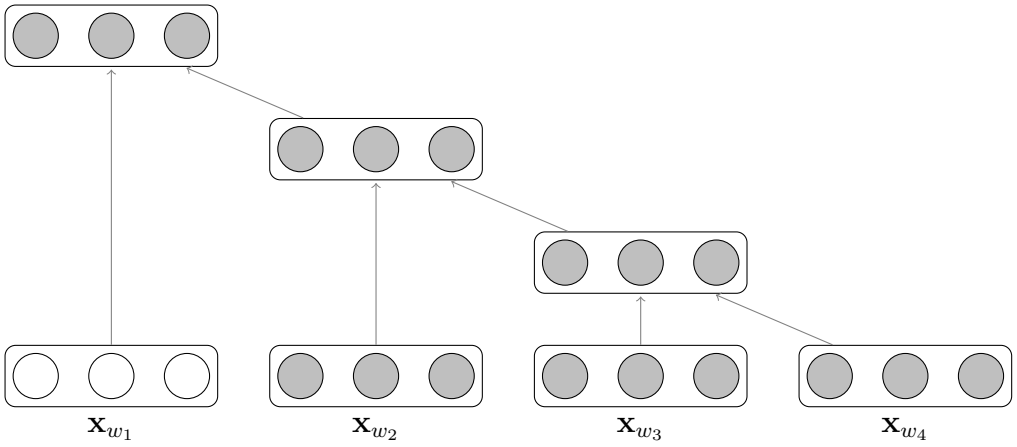


Figure 7.1: Encoding steps of the recursive autoencoder model. Note that all nodes are hidden as the input representations can be changed during training.

The model is semi-supervised as the target task can influence the representations. In addition to the usual autoencoder optimization objective of minimizing the *reconstruction error*, the model is also jointly trained to minimize the *classification error* made at each node in an external task.

Compared to models of compositionality in sentiment analysis, the RAE reflects both aspects of the clue model and the polarity shifter model. Each node in the tree has a polarity through the classification model (i.e., it serves as a clue). The representation, and thus the polarity, of non-terminal nodes is influenced by pairwise composition, which can be viewed as a shifter operation.

The basic representation of each word w_i is a randomly initialized vector $\mathbf{r}_{w_i}$ of dimensionality D.[19] The vectors are stored in a matrix $\mathbf{R}$ where every row represents one word. This representation is enhanced using an *embedding matrix* $\mathbf{L}$ of the same structure which is optimized during training. The final representation of a word w is obtained as the sum of its entries in $\mathbf{R}$ and $\mathbf{L}$,

$$\mathbf{x}_w = \mathbf{r}_w + \mathbf{l}_w. \tag{7.1}$$

[19] In the original publication, initialization with pre-trained NLM vectors resulted in a slight improvement.

The word vectors $\mathbf{x}_w$ make up the leaf nodes of the tree. Trees are then constructed by iteratively joining two adjacent nodes using an autoencoder with the usual two-layer architecture. Figure 7.1 illustrates the forward pass of the encoding layers. The encoding layer takes the feature vectors of two nodes, $\mathbf{n}_1$ and $\mathbf{n}_2$, and outputs a combined representation $\mathbf{r}$:

$$\begin{aligned} f^{(\text{enc})}([\mathbf{n}_1, \mathbf{n}_2]) &= h^{(\text{enc})}([\mathbf{n}_1, \mathbf{n}_2]\mathbf{W}^{(\text{enc})} + \mathbf{b}^{(\text{enc})}) \\ \mathbf{r} &= \frac{f^{(\text{enc})}([\mathbf{n}_1, \mathbf{n}_2])}{\|f^{(\text{enc})}([\mathbf{n}_1, \mathbf{n}_2])\|_2} \end{aligned}$$

Socher et al. (2011) argue that normalization of the representation vector serves to compensate for n-gram length. Subsequently, the reconstruction layer tries to reproduce the original inputs after normalization of the outputs:[20]

$$\begin{aligned} f^{(\text{rec})}(\mathbf{r}) &= h^{(\text{rec})}(\mathbf{r}\mathbf{W}^{(\text{rec})} + \mathbf{b}^{(\text{rec})}) \\ [\mathbf{n'}_1, \mathbf{n'}_2] &= \frac{f^{(\text{rec})}(\mathbf{r})}{\|f^{(\text{rec})}(\mathbf{r})\|_2} \end{aligned}$$

$h^{(\text{enc})}$ and $h^{(\text{rec})}$ are the (non-linear) activation functions of the respective layers. Socher et al. (2011) use $h^{(\text{enc})} = h^{(\text{rec})} = \tanh$.[21] $\mathbf{W}^{(\text{enc})}$ and $\mathbf{W}^{(\text{rec})}$ are the weight matrices, $\mathbf{b}^{(\text{enc})}$ and $\mathbf{b}^{(\text{rec})}$ the bias vectors of the respective layers. Note that D, the dimensionality of $\mathbf{r}$, needs to be the same as of $\mathbf{n}_1$ and $\mathbf{n}_2$ so that the autoencoder can be applied recursively.

The resulting output of the autoencoder serves as the representation of a new node that has the two encoded nodes as its children. The combination operation is carried out greedily by autoencoding the pair of nodes first that has minimal reconstruction error $E^{(\text{rec})}$. Reconstruction errors are calculated with an adaptation of squared error that penalizes errors at higher-level nodes less than lower-level nodes, which is controlled with a factor β.

Each node output is then used to predict the sentence label individually using

[20] Note that as shown in the source code, the output vectors $\mathbf{n}_1$ and $\mathbf{n}_2$ are normalized independently.

[21] Note that while the paper suggests a linear activation function for reconstruction in Equation 3, the code shows that tanh is being used for reconstruction as well.

a softmax layer

$$\mathbf{c} = f^{(\text{cl})}(\mathbf{r}) = \text{softmax}(\mathbf{r}\mathbf{W}^{(\text{cl})} + \mathbf{b}^{(\text{cl})}), \tag{7.2}$$

where $\mathbf{W}^{(\text{cl})}$ is the classification weight matrix, $\mathbf{b}^{(\text{cl})}$ the classification bias vector, and $\mathbf{r}$ the representation of a node. $\mathbf{c}$ is the prediction, a probability distribution over all possible classes. The error function $E^{(\text{cl})}$ on this classification task is cross-entropy.

There is a trade-off between two objectives that are minimized: the reconstruction error $E^{(\text{rec})}$ that specifies how well the resulting node represents the two children, and the classification error $E^{(\text{cl})}$ that measures how well the correct label of the sentence can be predicted from the information at the node. The two objectives are interpolated linearly with a coefficient α. Note that the embedding matrix $\mathbf{L}$ is optimized with both objectives as well, so the basic word representations can also be changed depending on their contribution to the supervised classification output.

The derivatives of the model are calculated with backpropagation through structure (Goller and Küchler, 1996). For batch optimization, arithmetic means of the errors $\bar{E}^{(\text{rec})}$ and $\bar{E}^{(\text{cl})}$ and the corresponding gradients are calculated over all nodes and all examples. Locally optimal weights are then found numerically through optimization with L-BFGS (Nocedal, 1980).

The way the model is applied to predict classes for unseen data differs from way used in training. Although a (node) classifier has been trained before, Socher et al. (2011) first perform a "feature extraction" step and then train a sentence classifier. In contrast to RAE training where the errors of the individual node predictions are averaged, they first calculate the arithmetic mean of all node features to get a single feature for the tree. Let $\mathbf{n}_1, \ldots, \mathbf{n}_k$ be the features of the k nodes in a tree. Feature extraction returns

$$\bar{\mathbf{n}} = \frac{\sum_i \mathbf{n}_i}{k}$$

as the combined vector representation of the tree. In the original implementation, the overall representation used for classification is the concatenation of the mean representation and the vector at the top node of the tree, $\mathbf{n}_{\text{TOP}}$:

$$\mathbf{n} = [\bar{\mathbf{n}}; \mathbf{n}_{\text{TOP}}]$$

We found that using the top feature did not improve the results significantly, so we will leave this feature out in our experiments. Finally, a single-layer softmax neural network is trained on this representation on the training data:

$$f^{(\mathrm{cl}_2)}(\mathbf{n}) = \mathrm{softmax}(\mathbf{n}\mathbf{W}^{(\mathrm{cl}_2)} + \mathbf{b}^{(\mathrm{cl}_2)}) \tag{7.3}$$

The model is of the same type as the softmax model used during training, but the input representations are different – node averages instead of individual nodes. Taking the mean of all nodes is a pooling operation. Similar operations have been applied successfully in NLP previously, for example by Collobert et al. (2011) who calculate the maximum over the dimensions of their representation vectors. In preliminary experiments, we found that replacing the mean operation by a maximum operation is a possible alternative but hurts the results slightly.

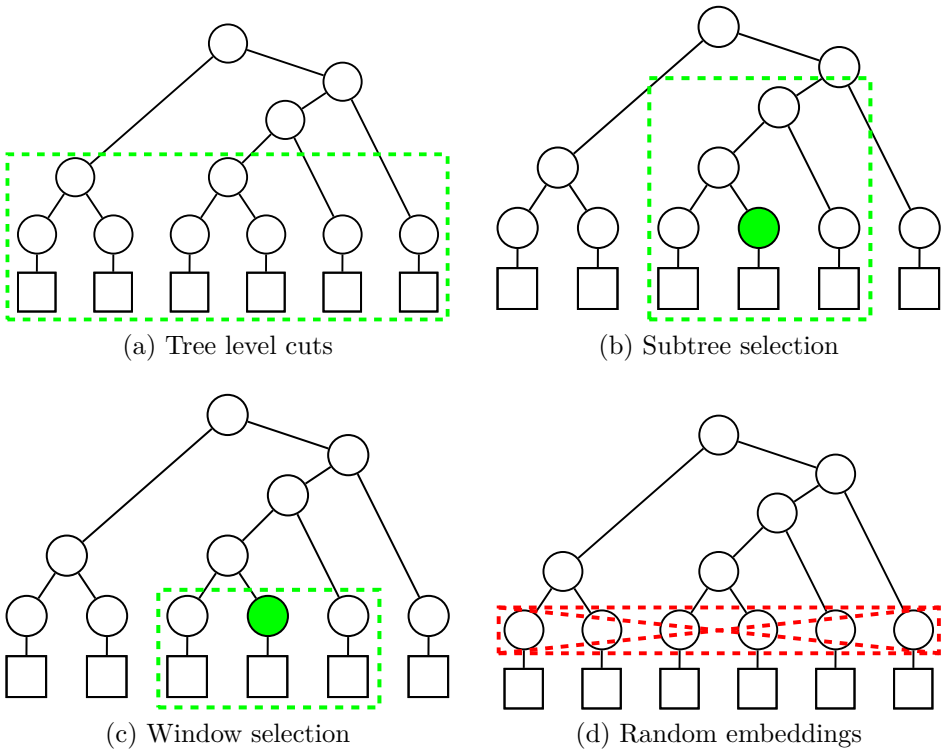
The RAE has several hyperparameters that need to be set. First, the interpolation weight of reconstruction and classification error α. Second, the penalty of higher-level node errors versus leaf errors β. In addition, the weight matrices of each autoencoder layer and the softmax classifier are regularized with the L_2 norm, giving rise to the regularization parameters $\sigma^{(\mathrm{enc})}$, $\sigma^{(\mathrm{rec})}$, and $\sigma^{(\mathrm{cl})}$.

7.3 Methods for Automatic Structural Simplification

In a complex model like the RAE it is difficult to see which parts are responsible for the results it achieves. In order to analyze the contributions of the automatically generated structures, we try to simplify them automatically. We will measure the performance of each model by evaluating it on a task. The result will then show whether the omission of the structure has an influence. We introduce four ways to structurally simplify RAE trees.

Tree Level Cuts

Our approach aims at determining the influence of higher level nodes in the tree. One straightforward operation that achieves this is a *level cut* where we remove



(a) Tree level cuts (b) Subtree selection

(c) Window selection (d) Random embeddings

Figure 7.2: Four methods for RAE simplification. Square nodes are random embeddings, first circled layers are embeddings. Green color indicates selection, color indicates omission.

nodes above a certain level.

We count levels starting at the leaves, the basic units for the RAE. All terminal nodes t are defined to have level $\ell(t) = 1$, and each non-terminal n with children $\langle c_1, c_2 \rangle$ has the level $\ell(n) = \max(\ell(c_1), \ell(c_2)) + 1$. We implement level cuts by first computing the full tree and then pruning away all nodes that have a level $\ell_n > \ell_{max}$. This process is sketched in Figure 7.2a which shows a cut at $\ell_{max} = 2$.

Subtree Selection

Another approach to simplification follows from the idea that not every word is important for sentiment classification, but rather that there is a region in the

sentence, centered around a clue, that is sufficient to recognize sentiment (Tu et al., 2012). A good example is the tree in Figure 7.3b. In order to predict the correct sentiment of the sentence, it is sufficient to analyze the second clause ("it never took off and always seemed static"). This way of simplification is orthogonal to level cuts. Level cuts reduce the amount of structure induced by the autoencoder but keep the complete input. Subtree selection reduces the input, but it makes use of a well-defined subtree instead of the full representation.

In order to select a region, we greedily select a *central word* (indicated in green in Figure 7.2b): We apply the softmax prediction function of the autoencoder (Equation 7.2) to each word in the sequence and pick the one with the lowest $E^{(\text{cl})}$ as the central word. For training examples, we compute the error for the gold class. For testing examples, we compute the score over all classes and select the word with the overall minimal error.

Starting from the central node, we traverse the tree towards the top node and select the largest subtree produced by the RAE whose top node n has a level of $\ell_n \leq \ell_{\max}$ (e.g., $\ell_{\max} = 3$ in Figure 7.2b).

Window Selection

The window approach is based on a similar motivation. Here, we again identify a central word as described above and take the representations of all words within a window of size w to either side as input to the classifier. All other words are ignored. We include the central word in the count. For example, $w = 3$ means that we take the two words to the left and to the right of the central word and drop everything else; and $w = 1$ only uses the central word and no context. Figure 7.2c shows an example for $w = 2$. In this approach, only the leaves (i.e., the random initializations plus the embeddings) of the trees are used. The rest of the tree is ignored.

Random Embeddings

To analyze the role of the learned embeddings, we also conduct an experiment where we train an RAE in the standard setup but do not add the embeddings during feature extraction. This means that Equation 7.1 simplifies to $\mathbf{x}_w = \mathbf{r}_w$.

This is illustrated in Figure 7.2d. This experiment will show (i) whether sufficient information can be encoded in the higher-level nodes of the tree to make useful predictions, and (ii) whether the RAE is robust to the loss of information.

7.4 Experiments

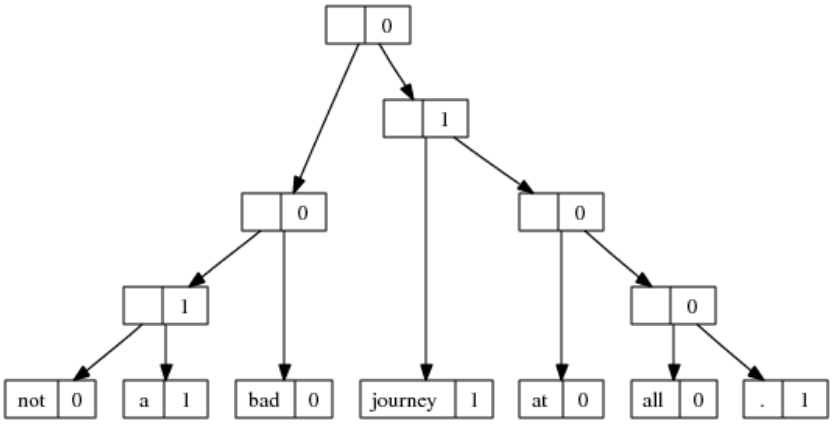
Error analysis proves to be difficult for automatically generated representations. In general, the dimensions produced by autoencoding in NLP applications cannot be interpreted easily. Therefore, we resort to empirical evaluation in the context of sentiment analysis.

The following experiments are designed to analyze the learned word and phrase representations, particularly with respect to the compositional effects of the model. We will conduct two types of experiments that will shed light on the compositionality of the RAE model. First, we let human annotators analyze the tree structures manually. Second, we run classification experiments with the automatic structural simplification methods introduced above.

7.4.1 Experimental Setup

In order to reproduce the original setup as closely as possible, we address the same task with dataset as (Socher et al., 2011): sentiment classification for the sentence polarity dataset by Pang and Lee (2005) which is described in Appendix A.2.1. It contains 10,662 sentences from movie reviews that were manually labeled as expressing positive or negative sentiment towards the movie.

We use the RAE implementation provided at `http://www.socher.org`. We set the hidden layer dimensionality to $D = 50$, the default setup in the code. All experiments are carried out on the predefined 90:10 training-testing split. We use accuracy as the evaluation measure as the class distribution in the dataset is balanced. We set the model parameters described in Section 7.2 as suggested by the authors.



(a) Sentence 1: *not a bad journey at all .*

Figure 7.3: Example RAE trees (cont. on the following pages). Leaf nodes contain the word that is encoded by them. All nodes contain the sentiment predicted using softmax, 1 being positive and 0 negative sentiment. We follow (Socher et al., 2011) and replace words for which no embedding is available by "*UNKNOWN*".

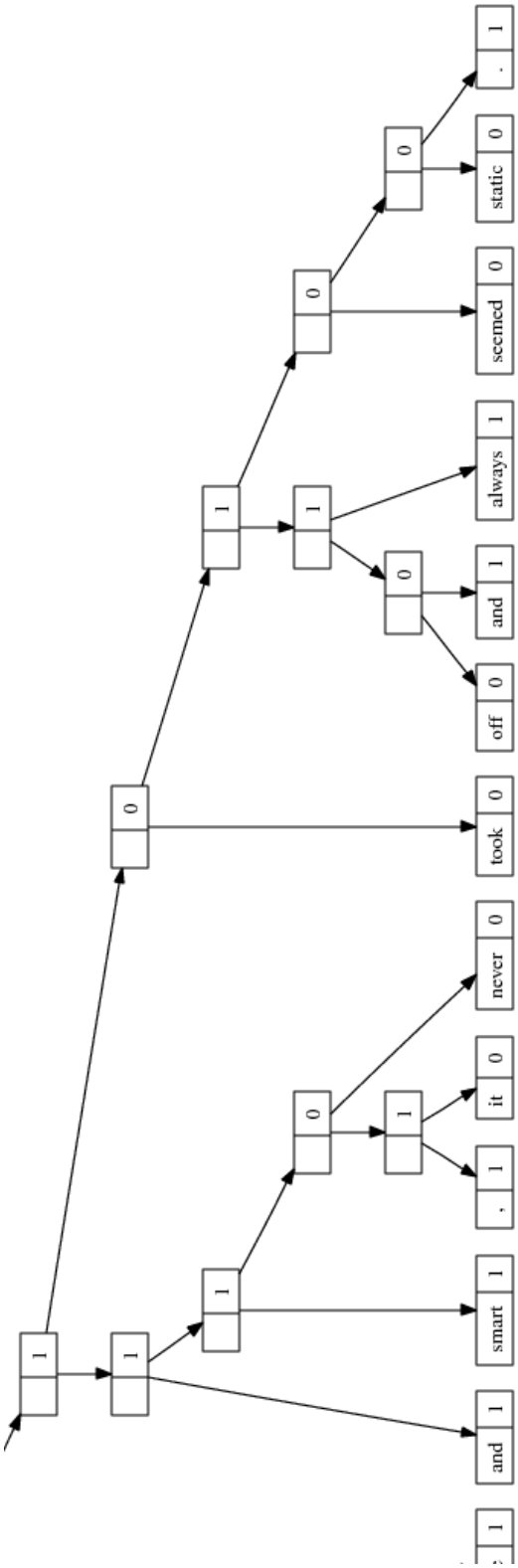
7.4.2 Human Evaluation

Linguists are used to tree structures representing syntactic properties of language. Naturally, it is tempting to adapt this view to RAE tree structures. In this section, we will provide manual analyses of their syntactic and semantic coherence conducted by human annotators.

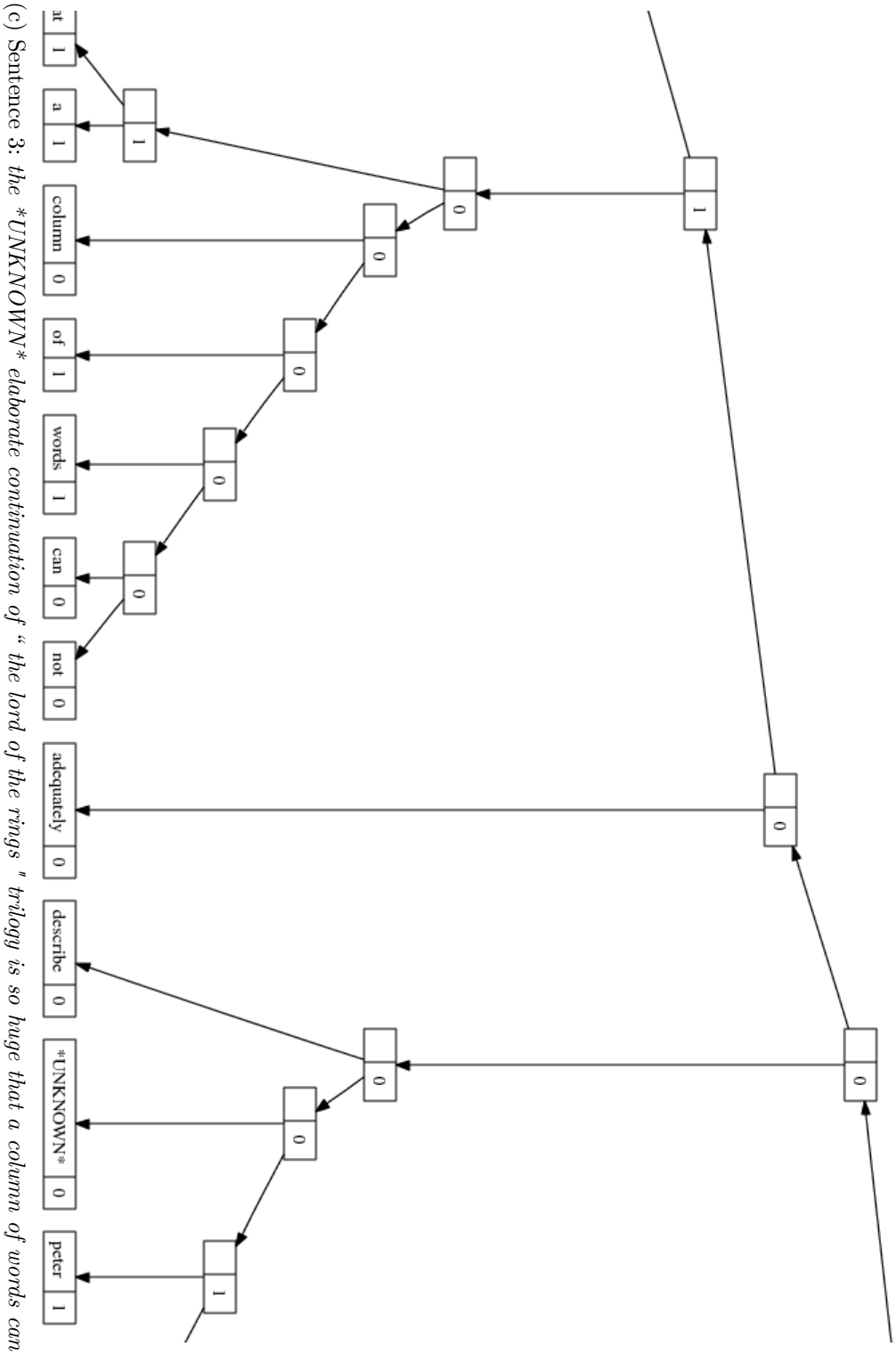
Syntactic Coherence

First, we will show that there is a large divergence between traditional syntactic trees as most theories of grammar would posit and the trees produced by RAE. We analyze two phenomena: coordinating *conjunction* and *negation*. We selected these particular phenomena as they are considered particularly important for sentiment analysis. Negation plays a key role as a polarity reverser. Conjunction is thought to be mostly sentiment-preserving.

Our annotation task is set up as follows. We randomly select 10 example sentences for each grammatical phenomenon, conjunction and negation, from the



(b) Sentence 2: *though everything might be literate and smart , it never took off and always seemed static .*



(c) Sentence 3: *the *UNKNOWN* elaborate continuation of " the lord of the rings " trilogy is so huge that a column of words can not adequately describe *UNKNOWN* peter *UNKNOWN* 's expanded vision of *UNKNOWN* . r . r . *UNKNOWN* 's *UNKNOWN* .

dataset. Candidates for random selection were identified through keyword search (*not* for negation, *and* and *or* for conjunction). We compute a tree representation for each sentence with the trained RAE. We then ask two human annotators to judge whether the parse of each examples was correct with respect to the phenomenon.

The human judges found unanimously that out of the 20 examples, none were correctly analyzed by the RAE. This result shows that it is unlikely that the RAE captures these syntactic structures. Of course, it may be argued that these phenomena could be particularly difficult to analyze automatically. While conjunctions are notoriously difficult even for supervised syntactic analysis models, negations are considered less problematic (McDonald, 2006; Collins, 1999). We also recognize that the RAE trained on sentiment data does not have the objective to learn syntactic structures, but to classify sentences. It may not be necessary to get the syntactic structure right in order to achieve this goal. However, we find that this result is important to understand the behavior of the model.

Error Analysis The example trees in Figure 7.3 illustrate these results. We will first take a look at examples for negation. In sentence 7.3a, the autoencoder used *not* at a low level in the tree, constituting a modification of *a*. Their joint representation is itself joined with *bad*. The correct analysis would use *not* as a modifier to a joint structure *a bad journey* where *bad* and *journey* are combined directly. Sentences 7.3b and 7.2c represent cases where the autoencoder wrongly introduced long distances between the negating and the negated phrase (*never* to *took off* and *not* to *describe*).

We now turn to conjunction. In sentence 7.3b, we find an instance of a coordination of two clauses. The clause *always seemed static* should receive a joint analysis and should then be modified by *and*. Instead, *and* is put in a subtree containing two words from each of the coordinated clauses.

One explanation for the resulting structures could be the greedy construction process. As RAEs are trained by joining the least error-prone combinations first, pairings of frequent words are common. For example, in sentence 7.3a, frequent words are joined first (*not* to *a*, and *all* to .). The most uncommon words are added last (*bad* and *journey*). In around 75% of all occurrences, periods are adjoined

directly to their neighbors, which is not desirable from a syntactic or semantic point of view (instead, punctuation is commonly attached to the top node).

Semantic Coherence

We will now analyze the behavior of RAEs from a semantic point of view. As we have seen earlier, sentiment analysis, particularly in a sparse data situation such as a sentence classification task, is thought to be able to benefit from a model capable of semantic composition.

In the clue model, sentiment is thought to be carried by a set of polarity-indicating expressions, for example adjectives like *great* or *awful* in combination with their close syntactic environment. In the polarity shifter model, the sentiment of a clue can then be modified by applying an intensifying or reversing construction. A simple check for whether RAEs are able to learn compositionality is to check the produced trees for instances of these modifiers – shifters and reversers – and see whether they behave as expected.

Shifters such as *very* or *little* are difficult to analyze as there is no straightforwardly quantifiable result that one would expect to occur in a binary classification setup. We would simply be unable to see a change in the label if the polarity is not changed. Therefore, we consider only reversers in this analysis. There is no consensus as to which words constitute the set of reversers. Often, reversing properties of a word are context-dependent (Polanyi and Zaenen, 2006; Kessler and Schütze, 2012). For this reason, we picked a small set of reversers with general applicability: *not*/*n't*, *no*, and *never*.

To check whether reversal occurs in a tree, we first calculate the classification decision by evaluating the softmax decision function at each node. We search the trees for occurrences of any reverser and check whether its sibling and its parent node are assigned opposite classes, which should be the case if the reverser was correctly applied. We find that reversal happens in only around 31% of all reverser occurrences.

We already noted that reversing is context-dependent, so we need human input to verify this result. From all trees containing reversers, we randomly selected 3 trees in which the reverser reverses sentiment and 7 trees in which the reverser

does not reverse sentiment (the goal being to mimic the 31% rate of reversals). We asked two human judges to check the examples for correct reversal. The judges unanimously found that only 3 out of 10 candidates are behaving correctly, confirming that reversal is very likely not a property captured correctly by the RAE model.

Error Analysis We again turn to Figure 7.3 for examples of errors. In sentence 7.3a, *not* does not reverse the polarity of *a* – which is correct reverser behavior. However, modifying *bad* with the resulting structure still does not reverse. The polarity of the whole sentence seems to be determined by the polarity of *journey* which gets reversed at the top node, leading to misclassification. In sentence 7.3b, *never* should reverse *took off*, yielding a negative sentiment overall. However at the point where the two phrases are joined, the topmost node depicted, positive sentiment is predicted.

We conclude that reverser behavior is not modeled well by the RAE. We reiterate that the effect of reversers is quite complex and in many contexts – e.g., "not awesome, but pretty good" – they do not simply reverse sentiment. However, the results indicate that the syntactic and semantic role of reversers is not modeled well in those cases where they act as simple reversers.

7.4.3 Automatic Structural Simplification

In the previous experiments, we showed that the structures generated by RAEs cannot be interpreted easily in terms of traditional linguistic insight from syntax and semantics. As mentioned above, this does not imply that the structures do not help in the classification task.[22] We will now turn to the empirical evaluation of the contributions of these structures in a sentence polarity prediction task. We apply the four simplification methods we introduced above.

Level Cuts during Feature Extraction

In the first experiment, we train the RAE to produce full trees and apply level cuts only in the feature extraction stage. We report accuracies in Table 7.1, column

[22] A well-known insight which is captured by a popular quote: "Airplanes don't flap their wings."

$\ell_{\max}/w$	EXTRACT	TRAIN+ EXTRACT	RAND- EMBED	SUBTREE	WINDOW
1	77.30	77.67	58.07	25.33	25.33
2	75.98	68.95	62.89	25.33	30.96
3	75.61	54.60	66.98	26.17	73.55
5	74.86	52.60	69.14	27.67	74.20
7	76.08	54.04	72.20	55.25	74.86
10	76.17	62.02	74.11	71.58	75.61
15	76.75	63.23	77.39	77.02	77.20
20	76.75	63.60	77.49	76.74	77.20
∞	76.75	77.24	77.49	77.20	77.21

Table 7.1: Accuracy using different structure simplification techniques

EXTRACT, for different values of $\ell_{\max}$. For reference, the trees produced by the RAE on the whole dataset have a mean height of 10 and a maximum height of 23.

First note that the best accuracy is achieved by cutting directly above the leaves, i.e. using no trees at all. Increasing the maximum level lowers accuracy significantly; when using $\ell_{\max} = 5$, for example, we lose over 2 percentage points compared to $\ell_{\max} = 0$. Only when $\ell_{\max}$ is increased further, accuracy recovers. This result suggests that the leaf representations – the embeddings – carry great weight in classification. It also suggests that the nodes close to the top of the tree contribute some stability.

Of course, one could argue that the level nodes are still being trained as a part of a larger structure and that this structure has a positive effect on the resulting representations. The following experiments will however show that this is not the case.

Level Cuts during Training

One could argue that using a fully-trained RAE to extract pruned trees is unfair since the model is still able to use the full information induced during training. In addition, most of the computation time is spent during RAE training as learning

such complex structures is very costly. In the following experiment, we perform level cuts during both training and feature extraction, using the same maximum level $\ell_{\max}$.

The column TRAIN+EXTRACT in Table 7.1 shows the results for this experiment. First, we observe that we get a well-performing model for $\ell_{\max} = 1$ in both RAE training and feature extraction. This result shows that the embeddings of the RAE are not affected by the structure since at this point, no structure is generated at all. Next, we can see that accuracy drops quickly as we introduce more levels and only recovers after raising the threshold to ∞, using full trees. A possible explanation for this phenomenon is that when enforcing low levels, there are also fewer training instances for the RAE and thus the resulting models are worse. Another possibility is that when full trees are constructed, all applications of the RAE depend on each other since errors are propagated through the structure. Thus, inconsistencies should be optimized away. However, there are fewer inconsistencies in lower-level cuts since the resulting subtrees are likely to be disconnected. This is further supported by the fact that mid-level nodes hurt the result even in the case where the trees are fully trained, which we demonstrated above.

These experiments show again that the best accuracy is achieved by a model that does not use the tree structures. Our conclusion from this evidence is that the strength of the RAE lies in the embeddings, not in the induced tree structure.

Random Embeddings

In order to demonstrate the significance of the embeddings, we again train full RAE trees but use the random representations as leaves for constructing trees during feature extraction. The results of this run are shown in Table 7.1, column RAND-EMBED.

Naturally, using only the random representations (at level $\ell_{\max} = 1$) we achieve low accuracy. Note that the results are still over chance level, an effect which may be caused by the randomly generated representations being equivalent to low-dimensional random indexing (Kanerva et al., 2000). Nevertheless, using higher-level tree representations successively increases accuracy to a similar level as observed in the previous experiment.

Our interpretation of this experiment is that while the trees seem to be able to create useful representations of the underlying words, these representations are redundant with respect to the embeddings. Combining both does not lead to improved classification accuracy. This property may however be helpful in cases where noisy input representations are expected.

Subtree Selection

We now turn to subtree selection. As stated previously, recent results suggest that sentiment mostly uses locally compositional structures, so it might be sufficient to use part of a sentence to classify the data. We perform subtree selection on trees from the fully trained RAE model.

Table 7.1, column SUBTREE shows the results for this experiment. We observe low accuracies for low $\ell_{\max}$. Only when using large contexts (recall that the maximum number of levels is around 23), the results become competitive. From this experiment, it is not clear whether the height of the trees or the size of the contexts – which grows with the height – is responsible for the gain. We will investigate this issue in the following experiment.

Error analysis shows that there are cases where an unsuitable central word is selected. For example, in sentence 7.3b, the selected central word is *might*, which occurs in the first part of the sentence that should actually be disregarded. In some cases, overfitting of the classifier will select a word that would commonly be considered to be non-polar, e.g., *and*, which is selected in sentence 7.2c.

A list of the most confidently classified words by the softmax node classifier is shown in Table 7.2. We find that this list of features mirrors in part the ones found early by active learning in Chapter 4. Overfitting still seems to be a problem (e.g., *jokes*), although some of the possible candidates for overfitting could in fact be considered clues (movies made for *cinema* are usually of superior quality compared to movies for *video* or *tv*).

Intuitively, there should be cases where sentiment is reversed, and thus the central word should have the opposite polarity of the sentence. However, we found that selecting based on the gold label during training improves the results slightly (probably because these cases are in the majority).

feature	sigmoid(**n**)	feature	sigmoid(**n**)
moving	0.9970	bad	0
culture	0.9956	dull	0
powerful	0.9954	boring	0.0019
enjoyable	0.9945	fails	0.0047
touching	0.9922	worst	0.0066
portrait	0.9920	jokes	0.0067
warm	0.9918	flat	0.0077
solid	0.9916	too	0.0102
cinema	0.9913	problem	0.0108
engrossing	0.9906	mediocre	0.0116
provides	0.9899	title	0.0121
captures	0.9878	tv	0.0123
thoughtful	0.9876	stupid	0.0130
wonderful	0.9875	routine	0.0132
rare	0.9849	lack	0.0137
beauty	0.9845	badly	0.0152
inventive	0.9831	mess	0.0152
heart	0.9802	video	0.0153
journey	0.9802	neither	0.0160
human	0.9793	waste	0.0188
always	0.9788	supposed	0.0202
heartwarming	0.9763	generic	0.0205
riveting	0.9742	pointless	0.0207
lively	0.9738	loud	0.0216
masterpiece	0.9717	silly	0.0228

Table 7.2: 25 most positive and most negative words determined by their softmax outputs (shown: positive probability) of the fully-trained RAE

Window Selection

As a last experiment, we concentrate on the word embeddings as they seem to be sufficient to achieve high accuracies on this task. This will also show whether subtrees or embedded words were responsible for the improvements with increasing tree height in the previous section.

We vary the window size w starting from 1, which is only the word itself, to ∞ (the maximum number of words any sentence has). The data has a mean sentence length of around 21 words and a maximum sentence length of 63 words.

Results are shown in Table 7.1, column WINDOW. While a small window size (1 or 2) produces bad results. This might be an effect of choosing the wrong central word, an issue we already examined more closely in the subtree selection case. Increasing window size, accuracy rises rapidly starting at $w = 3$. This means that we lose less than 4 percentage points compared to the best result when using only 5 words from each sentence. Taking a window of 15 words (in each direction, including the center, i.e., a total of 31 words) is necessary to match the best classifier. There are around 16% of the data that have more than 31 words, so it seems that for sentences of that length, there is sufficient material in the window that the model can exploit.

7.4.4 Discussion

We presented multiple experiments in which we simplified the RAE tree structures. All of these experiments point towards the embedding having the strongest influence on the end result. If embeddings are not used, accuracy drops almost to chance level. Using full trees and no embeddings seem to have the same effect as using only embeddings. However, using both representations together does not yield any improvement. This suggests that there is a large overlap between what the trees model and what the embeddings model – pointing towards a feature selection effect. Further, we saw evidence that the RAE trees produce robust representations. Even when using random inputs, the trees are able to extract useful information.

7.5 Summary

In this chapter, we presented linguistically-oriented analyses of a compositional sentiment model, the Semi-Supervised Recursive Autoencoder. We conducted two different experiments concerning the structures learned and generated by RAEs.

In a human annotation experiment, we showed that there is no simple way to interpret the structures produced by RAEs in terms of traditional linguistic theories of syntax and semantics. We then automatically reduced the tree structure in different ways and showed that for sentence-level sentiment analysis, the embedded words were sufficient to achieve state-of-the-art accuracy. Our experiments on window selection suggest that a structure as simple as a well-chosen subset of the words in a sentence produces a good model.

Overall, we conclude that structural simplifications are feasible for sentence-level sentiment prediction. Unfortunately, this result is disappointing regarding the compositionality of the model. It confirms the observations we also made in previous experiments in this thesis: We can achieve high sentiment classification performance when resorting to an instance of the clue model: averaging word representations.

8 Conclusions and Future Work

8.1 Contributions

In this thesis, we made several contributions to sentiment analysis. We will discuss them individually for each approach and then give an overall conclusion.

Redundancy of Clues In Chapter 4, we showed that active learning of document polarity in a noisy environment can be accomplished successfully. In our experiments using Amazon Mechanical Turk, active learning significantly outperformed a random selection baseline. Active learning leads to an overall cost reduction while achieving the same classification result. We found that the quality of annotations can be ensured by simple counter-spam measures. We analyzed the classifiers during the active learning process to shed some light on how features are learned. We showed that active learning can identify interesting features faster, reducing redundancies in the data.

In Chapter 5, we presented a novel approach to sentiment classification that allows for a seamless integration of word and document knowledge in a joint model. This approach addresses redundancy by directly labeling clues instead of documents. We proposed a unified graph representation of words and documents with a subsequent label propagation step through random walks. This model is efficient, using well-known eigenvector calculations. We showed that polarity induction with Personalized PageRank performs significantly better than the baseline using average polarity. We further bootstrap a supervised statistical classifier to fully exploit the word feature space from which we gain additional improvements. In these experiments, we identified the insensitivity of clues to topical contexts as one of the major drawbacks of the clue model.

Contextuality of Clues In Chapter 6, we addressed the problem of topical context through sentiment relevance, a novel concept for identifying useful content

for sentiment analysis. We created a new dataset that has sentiment relevance labels on the sentence level. We then presented two approaches to semi-supervised sentiment classification. The first approach, distant supervision, uses a database with domain information. The second, transfer learning, makes use of a conceptually related but different dataset of a larger size. We found that both approaches benefit from using features that offer some way of semantic generalization (lexical taxonomies and domain-specific named entity types). Both approaches apply a minimum cut graph model to introduce a sequence constraint that compensates for low-confidence misclassifications and class imbalance effects.

Chapter 7 presented research on the compositionality of sentiment. While compositional sentiment models are emerging, linguistic analysis of such models is lacking. We present experiments with the Semi-Supervised Recursive Autoencoder (RAE), a compositional neural network model that jointly learns to represent language and to make predictions. We show that the structures produced by the models are difficult to interpret linguistically for humans. We found that the structures can also be simplified greatly without any loss of accuracy, and that the automatically learned vector representations of words carry most of the weight. This leads to a more efficient and less complicated model.

General Remarks The first major underlying theme of our work was the role of clues and their interaction in sentiment prediction. We showed both the strengths and the weaknesses of the clue model. It performs well even in sparse data situations, but it can easily be distracted through topic effects. Therefore, it is both difficult to improve and difficult to beat. The clue model lends itself to approaches where clues and larger linguistic units are mixed, like a graph-based approach. Going beyond clues by modeling compositionality is a desired goal to that end. However, achieving composition without extensive supervision is not an easy task.

The second theme was efficiency. We showed that sentiment classification can be addressed with simple and efficient models, and that annotation cost efficiency can be improved by clever example selection. In addition, we were able to simplify an established neural network model for sentiment classification and showed that only some of its substructures are needed for high performance.

8.2 Future Work

Active Learning In Chapter 4, we applied only the most basic active learning techniques. It has however been shown (Settles, 2011) that active learning of feature labels can improve annotation speeds drastically. We could adapt this idea to different classifier types, such as joint models of words and documents as the one presented in Chapter 5, or even the compositional model in Chapter 7. The latter approach would be very appealing as the annotations of structural data for full supervision is particularly costly.

Feature Knowledge The experiments in Chapter 5, focused on in-domain classification in a single language. The graph framework is easily extensible to multiple languages. While multilingual sentiment classification has been addressed in the past, no work on multilingual feature knowledge integration exists. Domain adaptation is a challenge for this approach as words can have different polarities in different contexts. A possible approach to this problem would be unsupervised node splitting which has been applied successfully in syntactic parsing (Petrov and Klein, 2007), where multiple latent nodes could represent a word's polarity in different contexts.

Sentiment Relevance So far, we have explored sentiment relevance in a relatively limited scope. The experiments presented in Chapter 6 use movie review data. While this makes for a simple setup, there are phenomena that do not occur in this text genre. For example, one type of particularly non-relevant content are complaints about shipping and handling in product reviews. For this reason, sentiment relevance should be explored in the future with a more general perspective. This includes covering different domains which may give rise to a need for domain adaptation.

Our sentiment relevance classification approaches are based on two different semi-supervised learning approaches. The first obvious improvement is a combination of both approaches which might lead to a cumulative improvement. Further research could also explore different types of feature generalization. Currently, we perform generalization with external hand-crafted databases. This step could be

augmented by or replaced with an automatic representation learning approach, for example using state-of-the-art neural network models.

Compositional Sentiment Analysis In Chapter 7, we analyzed the Recursive Autoencoder for compositionality. This model was among the first to model sentiment through automatically learned task-specific hierarchical structures. In the meantime, various improvements have been suggested. As such research is often motivated from a machine learning perspective, it would be valuable to conduct similar linguistic analyses for the successors of the RAE.

General Remarks While we found that the clue model can be improved modularly through the approaches we proposed, a compositional approach is ultimately desired. However, we also showed that modeling compositionality in sentiment analysis is still a challenging task. Although various models have been proposed as alternatives to the ones presented in this thesis (e.g., Socher et al., 2013), these require a vast amount of supervision. Such approaches do not yet address complications like domain and topic changes. Linguistic phenomena such as irony, metaphors, and idiomatic expressions are further challenges. After all, sentiment is grounded in the mind of humans who, to express it, can employ all means of language.

A Resources

A.1 Tools

A.1.1 Stanford Classifier

The Stanford Classifier (Manning and Klein, 2003) implements a maximum entropy model (cf. Section 2.4.1). The model weights are learned using a Quasi-Newton optimizer. The input and output format for the classifier is one example per line. The tool is tailored for NLP, so it has some linguistic feature extraction capabilities, supporting string manipulation such as lowercasing as well as feature extraction through simple tokenization and n-gram extraction. Thus, it is possible to input raw texts. These and other options can be specified in a configuration file. Unless specified otherwise, we use the following extraction settings:

```
1.useSplitWords=true
1.splitWordsRegexp=\\s
useClassFeature=true
```

In this setup, we apply unigram extraction to the first column in the input file. Further, we use the class feature to model class distributions. We use the standard MaxEnt hyperparameter settings.

The `ColumnDataClassifer` class offers a command line tool for standalone use. The Stanford Classifier is implemented in Java and is freely available.[23]

A.1.2 HIPR

HIPR is a tool for finding the minimum cut of a graph. HIPR implements the push-relabel method of Cherkassky and Goldberg (1995). HIPR uses the DIMACS graph

[23] `http://nlp.stanford.edu/software/classifier.shtml`

format[24] for inputs and outputs. The tool is implemented in C and is available online.[25]

A.1.3 Mate Tagger and Parser

The Mate parser or Bohnet parser (Bohnet, 2010) is a graph-based dependency parser. The parser expects tokenized and sentence segmented inputs. It returns dependency trees in the CoNLL 2012 format.[26] It is available for multiple languages and comes with a part-of-speech tagger that has been cross-trained with the parsing model. The underlying statistical model of both the tagger and the parser is a perceptron trained with MIRA (Crammer et al., 2006). The tool is implemented in Java and freely available.[27] For our experiments, we used the development version available at the IMS.

A.2 Data

A.2.1 Text Corpora

Movie Corpus

The movie corpus (Pang et al., 2002) is a collection of 2000 movie reviews from the Internet Movie Database (IMDb). The corpus consists of 1000 positive and 1000 negative reviews. The data was collected at a time when IMDb had no consistent rating system, and so various star- and grade based systems had to be consolidated.

The corpus is supplied by the authors in a pre-processed form, in particular, tokenization and global lowercasing has been performed. Alternatively, the raw HTML files are supplied from which the full texts can be extracted. We opted for the latter form of the data as many NLP systems such as part-of-speech taggers, syntactic parsers, and named entity recognizers rely on capitalization features and some have special tokenization requirements. We extract the text from the HTML files with the goal of reproducing the structure of the published data as closely as

[24] http://dimacs.rutgers.edu/Challenges/
[25] http://www.igsystems.com/hipr/
[26] http://conll.cemantix.org/2012/data.html
[27] http://code.google.com/p/mate-tools/

	labeled		unlabeled	
Domain	# docs	# features	# docs	# features
books	2,000	481,173	4,465	1,022,277
DVD	2,000	470,995	3,586	847,556
electronics	2,000	309,629	5,681	868,336
kitchen	2,000	265,469	5,945	785,861

Table A.1: MDS statistics. Counts of documents and features in the labeled and unlabeled part of the dataset.

possible. We apply the same sentence splitting tool, MXTERMINATOR (Reynar and Ratnaparkhi, 1997).[28]

Sentence Polarity Dataset

Pang and Lee (2005) introduced the sentence polarity dataset. It consists of 10,662 sentences from movie reviews, each sentence being manually labeled as positive or negative. The dataset is balanced, consisting of 5,331 examples for each class. The data was collected automatically from Rotten Tomatoes,[29] a film review aggregation website that collects reviews from online sources such as newspapers and automatically calculates an overall score and verdict (*rotten* for negative, *fresh* for positive). For each source, Rotten Tomatoes lists a quote snippet that summarizes the review. The negative part of the dataset contains random snippets from rotten reviews, the positive part from fresh reviews.

Multi-Domain Sentiment Dataset

The Multi-Domain Sentiment Dataset Blitzer et al. (2007) contains product reviews from amazon.com. The dataset covers four different domains: books, DVDs, electronics, and kitchen. For each domain, a labeled set of 2000 reviews (1000 positive and 1000 negative) and an unlabeled set of varying size (between 3500 and 6000 reviews) is provided (Table A.1).

[28] `ftp://ftp.cis.upenn.edu/pub/adwait/jmx/jmx.tar.gz`
[29] `http://www.rottentomatoes.com/`

The publicly available version of the dataset is the result of heavy pre-processing as the data provided consists only of extracted features (unigram and bigram). Thus, we went back to the HTML versions as well and extracted the full text of the reviews.

The dataset was originally used for domain adaptation. However, the different domains provide interesting challenges themselves even in an in-domain setting, such as heavy variation in average document length (comparing for example books and kitchen).

Subjectivity Dataset

The subjectivity dataset (P&L, Pang and Lee, 2004) contains 10,000 sentences or snippets that were automatically gathered to reflect the authors' concept of subjectivity. The subjective data (*quote*) consists of summarizing quote snippets from Rotten Tomatoes. These quotes are usually highly subjective as they summarize the opinion expressed in a single review. There is a high overlap between the quote part and the sentence polarity dataset described above. Objective sentences (*plot*) are taken from IMDb plot summaries. The data contains a mixture of subjective and objective material. This issue is further discussed in Chapter 6. The authors supply 5,000 examples for each class.

The main version of the dataset has been preprocessed with the same techniques as the previous datasets by the same authors, however the raw HTML files for this dataset are incomplete. For this reason, we were unable to recover an unprocessed version.

A.2.2 Lexical Resources

Wilson Subjectivity and Polarity Clues

The subjectivity clues (Wilson et al., 2005b) are a lexical resource providing both subjectivity and polarity for English words and phrases. Each entry consists of a word (`word1`, which may be a stemmed form, indicated by `stemmed1`), the part of speech for each word (`pos1`), and the degree of subjectivity (`type`) and polarity of the entry (`priorpolarity`). The following listing contains some example entries.

```
type=strongsubj len=1 word1=sad pos1=adj stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=sad pos1=anypos stemmed1=y priorpolarity=negative
type=strongsubj len=1 word1=sadden pos1=verb stemmed1=y priorpolarity=negative
type=strongsubj len=1 word1=sadly pos1=anypos stemmed1=n priorpolarity=negative
type=strongsubj len=1 word1=sadness pos1=noun stemmed1=n priorpolarity=negative
type=weaksubj len=1 word1=safe pos1=adj stemmed1=n priorpolarity=positive
```

The way the lexicon was created is somewhat opaque as the authors state that it was "collected from a number of resources", including both manually and automatically labeled resources. The authors claim that a majority of the clues were collected using the extraction pattern technique from (Riloff and Wiebe, 2003). However, this technique is aimed at subjectivity detection, thus the origin of the polarity labels are still unclear

WordNet

WordNet (Miller, 1995) is a lexical semantic resource for English. Multiple words can form a synset, a set of words that may be used synonymously within a word sense. For example, the words *ignition*, *firing*, *lighting*, *kindling*, and *inflammation* form the synset with the ID 00378479, which is described by the gloss *the act of setting something on fire*. Additionally, hyponymy relations are defined between synsets, leading to a taxonomy structure. These relations may be used for generalization over objects (e.g., both *bread* and *cheese* are a type of *food*). We use WordNet 3.0 in this thesis. This version lists 117,798 nouns in 82,115 synsets. WordNet also contains words of other part-of-speech, such as adjectives and verbs, however they have much lower coverage and their taxonomies are relatively flat. Although WordNet contains more verbs than VerbNet, we cannot use it for our purposes as we require a pre-defined set of base classes. CoreLex, which we describe next, defines such classes for WordNet nouns but not for verbs.

CoreLex

CoreLex (Buitelaar, 1998) defines base classes (*basic types*) with respect to polysemy on WordNet. Each basic type is itself a node in WordNet. The author then defines CoreLex classes which are combinations of these types. In total, there are

type	synset id	polysemous class
abs	00012670	abstraction
act	00016649	act, human_action, human_activity
agt	00004473	causal_agent, cause, causal_agency
anm	00008030	animal, animate_being, beast, brute, creature, fauna
art	00011607	artifact, artefact
atr	00017586	attribute
cel	00003711	cell
chm	08907331	compound, chemical_compound
chm	08805286	chemical_element, element
com	00018599	communication
con	06465491	consequence, effect, outcome, result, upshot
ent	00002403	entity
evt	00016459	event
fod	00011263	food, nutrient
frm	00014558	shape, form
grb	05115837	biological_group
grp	00017008	group, grouping
grs	05119847	social_group
grs	05116476	people
hum	00004865	person, individual, someone, mortal, human, soul
lfr	00002728	life_form, organism, being, living_thing
lme	08322690	linear_measure, long_measure
loc	00014314	location
log	05450515	region
mea	00018966	measure, quantity, amount, quantum
mic	00740781	microorganism
nat	00009919	natural_object
nat	05715416	body_of_water, water
nat	05720524	land, dry_land, earth, ground, solid_ground, terra_firma
plt	00008894	plant, flora, plant_life
phm	00019295	phenomenon
pho	00009469	object, inanimate_object, physical_object
pos	00017394	possession
pro	08239006	process
prt	05650477	part, piece
psy	00012517	psychological_feature
qud	08310215	definite_quantity
qui	08310433	indefinite_quantity
rel	00017862	relation
spc	00015245	space
sta	00015437	state
sub	00010368	substance, matter
tme	09065837	time_period, period, period_of_time, amount_of_time
tme	09092294	time_unit, unit_of_time
tme	00014882	time

Table A.2: CoreLex basic types

class	basic types
agh	agt hum
pas	act atr pos
pas	atr pos
pas	atr pos rel
pas	atr pos sta
gsl	grs log
lor	log rel
ara	act art atr psy
ara	art atr
ara	art atr psy
ara	art atr sta
atp	atr psy
atp	atr psy sta
hum	grp hum
hum	grp hum nat
hum	grs hum
hum	hum
atr	atr
atr	atr pro
atr	atr pro sta
atr	atr sta

Table A.3: Selected CoreLex classes

44 basic types and 371 classes. The list of basic types is shown in Table A.2. The complete list of base class definitions is too long to reproduce here, so we selected the ones that are mentioned in the thesis (Table A.3). The complete list can be found online.[30]

VerbNet

VerbNet (Kipper et al., 2008) is a semantic lexicon of English verbs. VerbNet categorizes 5,726 verbs into 273 base classes. Each base class features a set of thematic roles, selectional restrictions on arguments and frames. We will not list all base classes here, as we find that the names are self-explanatory compared to the CoreLex classes. Instead we refer to the online reference.[31]

A.2.3 Other Datasets

IMDb Cast Database

The IMDb cast database[32] contains the complete cast listings of IMDb. We use the snapshot from 08/19/2011. The database is organized in several parts. The *actors* and *actresses* parts list for each actor/actress the characters they played in a movie. The file is organized by actor. Each line contains a single role. Character names are listed in brackets. In addition, some entries contain extra information, for example `Gordy Jr. (10 Years)` in the listing below. The following is an excerpt of the listing for the actor *Michael Cera*.

```
Cera, Michael    2010 MTV Movie Awards (2010) (TV)  [Himself]
                 Arrested Development (2012)  [George-Michael Bluth]
                 Comic-Con 2007 Live (2007) (TV)  [Himself]
                 Darling Darling (2005)  [Harold]  <1>
                 Exit 9 (2003) (TV)  [Charles]
                 Extreme Movie (2008)  [Fred]  <4>
                 Frequency (2000)  [Gordy Jr. (10 Years)]  <13>
```

[30] http://www.cs.brandeis.edu/~paulb/CoreLex/corelex.html
[31] http://verbs.colorado.edu/~mpalmer/projects/verbnet.html
[32] http://www.imdb.com/interfaces/

The director, screenwriter, and composer databases follows a similar structure, listing the films they participated in for each person. The following listing is taken from the entry for the director *Christopher Nolan*.

```
Nolan, Christopher (I)  Batman Begins (2005)
                        Doodlebug (1997)  (as Chris Nolan)
                        Following (1998)
                        Inception (2010)
                        Insomnia (2002/I)
                        Memento (2000)
                        The Dark Knight (2008)
                        The Dark Knight Rises (2012)
                        The Prestige (2006)
```

Dict.cc Bilingual Lexicon

The `dict.cc` dictionary[33] is a user-edited online dictionary. We use the German-English version dated 05/05/2008. In this version, not all entries are tagged with part of speech. Generally, verbs are untagged, so we heuristically assume that all entries whose English forms begin with *to* are verbs. This pattern will however overgenerate, for example by matching idioms that start with to. Counting the tags, we get a total of 50,748 adjectives, 230,674 nouns, 11,984 adverbs, 48,634 verbs and a remainder of 75,458 uncategorized entries. The following listing contains some example entries.

```
abdrehen [z.B. Thermostat] :: to turn down
Abdrehen {n} :: turning
Abdrehen {n} [Bearbeiten] :: lathe machining
ab dreizehn Jahre :: thirteen years of age and over [Br.]
abdriften :: to make leeway
abdriften :: to space out [mentally]
Abdriftkorrektur {f} :: drift correction
```

[33] `http://www.dict.cc/`

Bibliography

Abney, S. (2010). *Semisupervised learning for computational linguistics*. CRC Press.

Akkaya, C., Conrad, A., Wiebe, J., and Mihalcea, R. (2010). Amazon Mechanical Turk for subjectivity word sense disambiguation. In *Proceedings of the NAACL HLT 2010 Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*, pages 195–203.

Artstein, R. and Poesio, M. (2008). Inter-coder agreement for computational linguistics. *Computational Linguistics*, 34(4):555–596.

Baker, M. and Wurgler, J. (2007). Investor sentiment in the stock market. *Journal of Economic Perspectives*, 21(2):129–152.

Banfield, A. (1982). *Unspeakable sentences: Narration and representation in the language of fiction*. Routledge & Kegan Paul.

Bengio, Y. (2009). Learning deep architectures for AI. *Foundations and Trends in Machine Learning*, 2(1):1–127.

Bengio, Y., Courville, A., and Vincent, P. (2013). Representation learning: A review and new perspectives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1798–1828.

Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003). A neural probabilistic language model. *Journal of Machine Learning Research (JMLR)*, 3:1137–1155.

Berger, A. L., Pietra, V. J. D., and Pietra, S. A. D. (1996). A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71.

Bianchini, M., Gori, M., and Scarselli, F. (2005). Inside PageRank. *ACM Transactions on Internet Technology (TOIT)*, 5(1):92–128.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer.

Blitzer, J., Dredze, M., and Pereira, F. (2007). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics (ACL)*, pages 440–447.

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling)*, pages 89–97.

Bourlard, H. and Kamp, Y. (1988). Auto-association by multilayer perceptrons and singular value decomposition. *Biological Cybernetics*, 59(4-5):291–294.

Brew, A., Greene, D., and Cunningham, P. (2010). Using crowdsourcing and active learning to track sentiment in online media. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI)*, pages 145–150.

Brinker, K. (2003). Incorporating diversity in active learning with support vector machines. In *Proceedings of the Twentieth International Conference on Machine Learning (ICML)*, pages 59–66.

Buitelaar, P. (1998). *CoreLex: Systematic polysemy and underspecification*. PhD thesis, Brandeis University.

Cherkassky, B. V. and Goldberg, A. V. (1995). On implementing push-relabel method for the maximum flow problem. In *Integer Programming and Combinatorial Optimization*, volume 920, pages 157–171. Springer.

Choi, Y. and Cardie, C. (2008). Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 793–801.

Choi, Y. and Cardie, C. (2009). Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 590–598.

Chomsky, N. (1965). *Aspects of the Theory of Syntax*. MIT Press.

Cleveland, W. S. and Devlin, S. J. (1988). Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association*, 83(403):596–610.

Collins, M. (1999). *Head-driven statistical models for natural language parsing*. PhD thesis, University of Pennsylvania.

Collobert, R. and Weston, J. (2008). A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning (ICML)*, pages 160–167.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, 12:2493–2537.

Crammer, K., Dekel, O., Keshet, J., Shalev-Shwartz, S., and Singer, Y. (2006). Online passive-aggressive algorithms. *Journal of Machine Learning Research (JMLR)*, 7:551–585.

Dahl, G. E., Yu, D., Deng, L., and Acero, A. (2012). Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *IEEE Transactions on Audio, Speech, and Language Processing*, 20(1):30–42.

Dasgupta, S. and Ng, V. (2009). Mine the easy, classify the hard: A semi-supervised approach to automatic sentiment classification. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (AFNLP)*, pages 701–709.

Deselaers, T., Hasan, S., Bender, O., and Ney, H. (2009). A deep learning approach to machine transliteration. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 233–241.

Diestel, R. (2000). *Graph Theory*. Springer.

Difallah, D. E., Demartini, G., and Cudré-Mauroux, P. (2013). Pick-a-crowd: Tell me what you like, and i'll tell you what to do. In *Proceedings of the 22nd international conference on World Wide Web (WWW)*, pages 367–374.

Ding, X., Liu, B., and Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM)*, pages 231–240.

Eguchi, K. and Lavrenko, V. (2006). Sentiment retrieval using generative models. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 345–354.

Fahrni, A. and Klenner, M. (2008). Old wine or warm beer: Target-specific sentiment analysis of adjectives. In *Proceedings of the Symposium on Affective Language in Human and Machine (AISB)*, pages 60–63.

Ford, L. R. and Fulkerson, D. R. (1956). Maximal flow through a network. *Canadian Journal of Mathematics*, 8(3):399–404.

Fort, K., Adda, G., and Cohen, K. B. (2011). Amazon mechanical turk: Gold mine or coal mine? *Computational Linguistics*, 37(2):413–420.

Foster, J., Cetinoglu, O., Wagner, J., Le Roux, J., Nivre, J., Hogan, D., and van Genabith, J. (2011). From news to comment: Resources and benchmarks for parsing the language of Web 2.0. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*, pages 893–901.

Gamon, M. (2004). Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling)*, pages 841–847.

Ganu, G., Elhadad, N., and Marian, A. (2009). Beyond the stars: Improving rating predictions using review text content. In *Proceedings of the Twelfth International Workshop on the Web and Databases (WebDB)*.

Glorot, X., Bordes, A., and Bengio, Y. (2011). Domain adaptation for large-scale sentiment classification: A deep learning approach. In *Proceedings of the 28th International Conference on Machine Learning (ICML)*, pages 513–520.

Go, A., Bhayani, R., and Huang, L. (2009). Twitter sentiment classification using distant supervision. CS224N Project Report, Stanford.

Goller, C. and Küchler, A. (1996). Learning task-dependent distributed representations by backpropagation through structure. In *IEEE International Conference on Neural Networks*, pages 347–352.

Grolmusz, V. (2012). A note on the PageRank of undirected graphs. *CoRR*, abs/1205.1960.

Gwet, K. L. (2008). Computing inter-rater reliability and its variance in the presence of high agreement. *British Journal of Mathematical and Statistical Psychology*, 61(1):29–48.

Haertel, R., Felt, P., Ringger, E. K., and Seppi, K. (2010). Parallel active learning: Eliminating wait time with minimal staleness. In *Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing*, pages 33–41.

Hassan, A. and Radev, D. R. (2010). Identifying text polarity using random walks. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 395–403. Association for Computational Linguistics.

Hastie, T., Tibshirani, R., and Friedman, J. J. H. (2001). *The elements of statistical learning*. Springer.

Hatzivassiloglou, V. and McKeown, K. R. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 8th Conference on European Chapter of the Association for Computational Linguistics (EACL)*, pages 174–181.

Haveliwala, T. H. (2003). Topic-Sensitive PageRank: A context-sensitive ranking algorithm for web search. *IEEE Transactions on Knowledge and Data Engineering*, 15(4):784–796.

He, Y. (2010). Learning sentiment classification model from labeled features. In *Proceedings of the 19th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1685–1688.

He, Z., Liu, S., Li, M., Zhou, M., Zhang, L., and Wang, H. (2013). Learning entity representation for entity disambiguation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, pages 30–34.

Hearst, M. A. (1992). Direction-based text interpretation as an information access refinement. In *Text-based Intelligent Systems*, pages 257–274. L. Erlbaum Associates Inc.

Heerschop, B., van Iterson, P., Hogenboom, A., Frasincar, F., and Kaymak, U. (2011). Analyzing sentiment in a large set of web data while accounting for negation. In *Proceedings of the 7th Atlantic Web Intelligence Conference (AWIC)*, pages 195–205.

Hinton, G. E. and Salakhutdinov, R. R. (2006). Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257.

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 (CIKM)*, pages 168–177.

Ipeirotis, P. G., Provost, F., and Wang, J. (2010). Quality management on amazon mechanical turk. In *Proceedings of the ACM SIGKDD Workshop on Human Computation*, pages 64–67.

Kanayama, H. and Nasukawa, T. (2006). Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 355–363.

Kandasamy, D. M., Curtis, K., Fox, A., and Patterson, D. (2012). Diversity within the crowd. In *Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work Companion (CSCW)*, pages 115–118.

Kanerva, P., Kristofersson, J., and Holst, A. (2000). Random indexing of text samples for latent semantic analysis. In *Proceedings of the 22nd Annual Conference of the Cognitive Science Society*, page 1036.

Kessler, W. and Schütze, H. (2012). Classification of inconsistent sentiment words using syntactic constructions. In *Proceedings of the 28th International Conference on Computational Linguistics (Coling)*, pages 569–578.

Kim, S.-M. and Hovy, E. (2004). Determining the sentiment of opinions. In *Proceedings of the 20th International Conference on Computational Linguistics (Coling)*, pages 1367–1373.

Kipper, K., Korhonen, A., Ryant, N., and Palmer, M. (2008). A large-scale classification of English verbs. *Language Resources and Evaluation*, 42(1):21–40.

Koppel, M. and Schler, J. (2006). The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2):100–109.

Krizhevsky, A., Sutskever, I., and Hinton, G. (2012). ImageNet classification with deep convolutional neural networks. In *Proceedings of Advances in Neural Information Processing Systems (NIPS) 25*, pages 1106–1114.

Lakoff, G. and Johnson, M. (1980). *Metaphors we live by*. University of Chicago Press.

Laws, F. (2013). *Effective active learning for complex natural language processing tasks*. PhD thesis, Universität Stuttgart.

Laws, F., Scheible, C., and Schütze, H. (2011). Active learning with Amazon Mechanical Turk. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1546–1556.

LeCun, Y., Bottou, L., Bengio, Y., and Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 3–12.

Li, S., Ju, S., Zhou, G., and Li, X. (2012). Active learning for imbalanced sentiment classification. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing (EMNLP) and Computational Natural Language Learning (CoNLL)*, pages 139–148.

Lin, C. and He, Y. (2009). Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM)*, pages 375–384.

Linke, A., Markus, N., and Portmann, P. (1994). *Studienbuch Linguistik*. Niemeyer.

Liu, B. (2010). Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing*. CRC Press.

Liu, B. (2012). Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167.

Manning, C. and Klein, D. (2003). Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL): Tutorials - Volume 5*, page 8.

Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Marcus, M. P., Marcinkiewicz, M. A., and Santorini, B. (1993). Building a large annotated corpus of english: The Penn Treebank. *Computational Linguistics*, 19(2):313–330.

Matsumoto, S., Takamura, H., and Okumura, M. (2005). Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, pages 301–311.

McCallum, A. (2002). MALLET: A machine learning for language toolkit. `http://mallet.cs.umass.edu`.

McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The Bulletin of Mathematical Biophysics*, 5(4):115–133.

McDonald, R. (2006). *Discriminative learning and spanning tree algorithms for dependency parsing*. PhD thesis, University of Pennsylvania.

Melville, P., Gryc, W., and Lawrence, R. D. (2009). Sentiment analysis of blogs by combining lexical knowledge with text classification. In *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (CIKM)*, pages 1275–1284.

Mihalcea, R. and Radev, D. (2011). *Graph-based natural language processing and information retrieval*. Cambridge University Press.

Miller, G. A. (1995). Wordnet: A lexical database for english. *Communications of the Association for Computing Machinery (ACM)*, 38(11):39–41.

Mintz, M., Bills, S., Snow, R., and Jurafsky, D. (2009). Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics (ACL) and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (AFNLP)*, pages 1003–1011.

Mitchell, T. M. (1997). *Machine learning*. McGraw Hill.

Moschitti, A. (2006). Making tree kernels practical for natural language learning. In *Proceedings of the 7th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 113–120.

Mount, J. (2011). The equivalence of logistic regression and maximum entropy models. Technical report, Win-Vector LLC Data Science Consulting.

Mullen, T. and Collier, N. (2004). Incorporating topic information into semantic analysis models. In *The Companion Volume to the Proceedings of 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 190–193.

Nakagawa, T., Inui, K., and Kurohashi, S. (2010). Dependency tree-based sentiment classification using crfs with hidden variables. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 786–794.

Ng, V., Dasgupta, S., and Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the Conference on Computational Linguistics (Coling) and Association for Computational Linguistics (ACL) Main Conference Poster Sessions*, pages 611–618.

Nigam, K., Lafferty, J., and McCallum, A. (1999). Using maximum entropy for text classification. In *Proceedings of the IJCAI-99 Workshop on Machine Learning for Information Filtering*, pages 61–67.

Nocedal, J. (1980). Updating quasi-newton matrices with limited storage. *Mathematics of Computation*, 35(151):773–782.

Noreen, E. W. (1989). *Computer Intensive Methods for Hypothesis Testing: An Introduction*. Wiley.

Norvig, P. (2011). On Chomsky and the two cultures of statistical learning. Online essay. `http://norvig.com/chomsky.html`.

Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab.

Pang, B. and Lee, L. (2004). A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL), Main Volume*, pages 271–278.

Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 115–124.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86.

Pearson, K. (1905). The problem of the random walk. *Nature*, 72(1865):294.

Petrov, S. and Klein, D. (2007). Learning and inference for hierarchically split pcfgs. In *Proceedings of the 22nd National Conference on Artificial Intelligence (NCAI)*, pages 1663–1666.

Polanyi, L. and Zaenen, A. (2006). Contextual valence shifters. In *Computing Attitude and Affect in Text: Theory and Applications*, volume 20, pages 1–10. Springer.

Pustejovsky, J. and Stubbs, A. (2012). *Natural Language Annotation for Machine Learning – a Guide to Corpus-Building for Applications*. O'Reilly.

Ratnaparkhi, A. (1996). A maximum entropy model for part-of-speech tagging. In *Proceedings of the 1996 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 133–142.

Reynar, J. C. and Ratnaparkhi, A. (1997). A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*, pages 16–19.

Riloff, E. and Wiebe, J. (2003). Learning extraction patterns for subjective expressions. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 105–112.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65(6):386–408.

Rummelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(9):533–535.

Salton, G. and McGill, M. J. (1986). *Introduction to Modern Information Retrieval*. McGraw-Hill.

Scheffer, T., Decomain, C., and Wrobel, S. (2001). Active hidden markov models for information extraction. In *Proceedings of the 4th International Conference on Advances in Intelligent Data Analysis (IDA)*, pages 309–318.

Scheible, C. (2010). Sentiment translation through lexicon induction. In *Proceedings of the ACL 2010 Student Research Workshop (SRW)*, pages 25–30.

Scheible, C., Laws, F., Michelbacher, L., and Schütze, H. (2010). Sentiment translation through multi-edge graphs. In *Proceedings of the 26th International Conference on Computational Linguistics (Coling): Posters*, pages 1104–1112.

Scheible, C. and Schütze, H. (2012). Bootstrapping sentiment labels for unannotated documents with Polarity PageRank. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC)*.

Scheible, C. and Schütze, H. (2013a). Cutting recursive autoencoder trees. In *Proceedings of the International Conference on Learning Representations (ICLR) 2013*.

Scheible, C. and Schütze, H. (2013b). Sentiment relevance. In *Proceedings of the 51st Meeting of the Association for Computational Linguistics (ACL)*, pages 954–963.

Scherer, K. R. (2003). Vocal communication of emotion: A review of research paradigms. *Speech communication*, 40(1):227–256.

Settles, B. (2009). Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison.

Settles, B. (2011). Closing the loop: Fast, interactive semi-supervised annotation with queries on features and instances. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1467–1478.

Shannon, C. E. (1948). A mathematical theory of communication. *The Bell System Technical Journal*, 27:397–423.

Smith, N. A. (2011). *Linguistic Structure Prediction*. Morgan and Claypool.

Snow, R., O'Connor, B., Jurafsky, D., and Ng, A. (2008). Cheap and fast – but is it good? evaluating non-expert annotations for natural language tasks. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 254–263.

Socher, R., Manning, C., and Ng, A. (2010). Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop*.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 151–161.

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642.

Solomon, R. C. (1977). The logic of emotion. *Noûs*, 11(1):41–49.

Spertus, E. (1997). Smokey: Automatic recognition of hostile messages. In *Proceedings of the 14th National Conference on Artificial Intelligence (AAAI) and 9th Innovative Applications of Artificial Intelligence Conference (IAAI)*, pages 1058–1065.

Su, F. and Markert, K. (2008). Eliciting subjectivity and polarity judgements on word senses. In *Proceedings of the Workshop on Human Judgements in Computational Linguistics*, pages 42–50.

Surdeanu, M., McClosky, D., Tibshirani, J., Bauer, J., Chang, A. X., Spitkovsky, V. I., and Manning, C. D. (2010). A simple distant supervision approach for the TAC-KBP slot filling task. In *Proceedings of the Third Text Analysis Conference (TAC 2010)*.

Taboada, M., Brooke, J., and Stede, M. (2009). Genre-based paragraph classification for sentiment analysis. In *Proceedings of the SIGDIAL 2009 Conference: The 10th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 62–70.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.

Täckström, O. and McDonald, R. (2011). Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval (ECIR)*, pages 368–374.

Tan, S. and Cheng, X. (2009). Improving SCL model for sentiment-transfer learning. In *Proceedings of Human Language Technologies (HLT): The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics (ACL), Companion Volume: Short Papers*, pages 181–184.

Thrun, S. (1996). Is learning the n-th thing any easier than learning the first? In *Proceedings of Advances in Neural Information Processing Systems (NIPS) 9*, pages 640–646.

Titov, I. and McDonald, R. (2008). A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL): Human Language Technology (HLT)*, pages 308–316.

Tu, Z., He, Y., Foster, J., van Genabith, J., Liu, Q., and Lin, S. (2012). Identifying high-impact sub-structures for convolution kernels in document-level sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, pages 338–343.

Turney, P. (2002). Thumbs up or thumbs down? Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 417–424.

Turtle, H. and Croft, W. B. (1991). Evaluation of an inference network-based retrieval model. *ACM Transactions on Information Systems*, 9(3):187–222.

Wang, S. and Manning, C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, pages 90–94.

Wellman, M. P. and Henrion, M. (1993). Explaining "explaining away". *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(3):287–292.

Wiebe, J. and Mihalcea, R. (2006). Word sense and subjectivity. In *Proceedings of the 21st International Conference on Computational Linguistics (Coling) and 44th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1065–1072.

Wiebe, J. and Riloff, E. (2005). Creating subjective and objective sentence classifiers from unannotated texts. *Computational Linguistics and Intelligent Text Processing*, pages 486–497.

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. (1990). *Recognizing Subjective Sentences: A Computational Investigation of Narrative Text*. PhD thesis, State University of New York at Buffalo.

Wiegand, M., Balahur, A., Roth, B., Klakow, D., and Montoyo, A. (2010). A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68.

Wilson, T., Hoffmann, P., Somasundaran, S., Kessler, J., Wiebe, J., Choi, Y., Cardie, C., Riloff, E., and Patwardhan, S. (2005a). OpinionFinder: A system for subjectivity analysis. In *Proceedings of the Conference on Human Language Technology (HLT) and Empirical Methods in Natural Language Processing (EMNLP): Interactive Demonstrations*, pages 34–35.

Wilson, T. and Wiebe, J. (2003). Annotating opinions in the world press. In *Proceedings of the 4th SIGdial Workshop on Discourse and Dialogue*, pages 13–22.

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 (HLT) and Empirical Methods in Natural Language Processing (EMNLP)*, pages 347–354.

Yang, B. and Cardie, C. (2013). Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1640–1649.

Yeh, A. (2000). More accurate tests for the statistical significance of result differences. In *Proceedings of the 18th Conference on Computational Linguistics (Coling)*, pages 947–953.

Yessenalina, A. and Cardie, C. (2011). Compositional matrix-space models for sentiment analysis. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 172–182.

Yessenalina, A., Yue, Y., and Cardie, C. (2010). Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1046–1056.

Zaidan, O., Eisner, J., and Piatko, C. (2007). Using "annotator rationales" to improve machine learning for text categorization. In *Proceedings of the 2007 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, pages 260–267.

Zhang, L., Liu, B., Lim, S. H., and O'Brien-Strain, E. (2010). Extracting and ranking product features in opinion documents. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling): Posters*, pages 1462–1470.

Zhuang, L., Jing, F., and Zhu, X.-Y. (2006). Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management (CIKM)*, pages 43–50.

Zipf, G. K. (1935). *The psycho-biology of language.* Houghton Mifflin.