Weakly Supervised Learning for Compositional Sentiment Recognition

Master's thesis

Michael Haas
Registration ID: 2775430
haas@cl.uni-heidelberg.de

Thesis submitted on 2014-12-22 in partial satisfaction of the requirements for the degree of Master of Arts in Computational Linguistics.

Heidelberg University
Faculty of Modern Languages
Institute of Computational Linguistics

Supervisor and First Examiner:
Dr. Yannick Versley
Second Examiner:
Prof. Dr. Anette Frank

EIDESSTATTLICHE ERKLÄRUNG

Hiermit versichere ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe, dass alle Stellen der Arbeit, die wörtlich oder sinngemäß aus anderen Quellen übernommen wurden, als solche kenntlich gemacht sind und dass die Arbeit in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegt wurde.

In Zusammenarbeit mit meinem Betreuer, Dr. Yannick Versley, entstand ein Papier basierend auf Teilen meiner Master-Arbeit. Dieses Papier wurde für die Konferenz NAACL-HLT 2015 unter dem Titel *Subsentential Sentiment Analysis on a Shoestring: A Crosslingual Analysis of Compositional Classification* eingereicht.

Heidelberg, den 22. Dezember, 2014 Michael Haas

STATUTORY DECLARATION

I declare that I have authored this thesis independently, that I have not used other than the declared sources / resources, that I have explicitly marked all material which has been quoted either literally or by content from the used sources and that this work has not been submitted, either in part or whole, for a degree at this or any other University.

A conference submission based on parts of this thesis has been prepared in cooperation with my supervisor, Dr. Yannick Versley. The paper titled *Subsentential Sentiment Analysis on a Shoestring: A Crosslingual Analysis of Compositional Classification* has been submitted to the NAACL-HLT 2015 conference.

Heidelberg, 22th of December, 2014 Michael Haas

Contents

1. Deutsche Zusammenfassung 5

1.1. Sentiment-Analyse . . . 5

1.2. Daten . . . 5

1.3. Verfahren . . . 6

1.3.1. Lexikon-Induktion . . . 6

1.3.2. Projektion . . . 6

1.3.3. RNTN und Verbesserungen . . . 6

1.3.4. Kombinationen . . . 7

1.4. Ergebnisse . . . 7

2. Introduction 9

2.1. The Problem . . . 10

2.2. Related Work . . . 10

2.2.1. Sentiment Analysis . . . 10

2.2.2. Cross-Lingual Sentiment Analysis . . . 23

2.2.3. Ensemble Learning . . . 27

2.2.4. German Sentiment Analysis - State of the Art . . . 30

2.2.5. Summary . . . 35

2.3. My Approach . . . 35

3. Resources 37

3.1. The Heidelberg Sentiment Treebank . . . 37

3.1.1. Annotation with Crowdflower . . . 39

4. Components 49

4.1. Sentiment lexicon extraction from weakly-labeled near-domain reviews . . 50

4.1.1. Data Acquisition . . . 50

4.1.2. Domain Adaption . . . 50

4.1.3. Method . . . 53

4.1.4. Evaluation . . . 56

4.1.5. Rating prediction task on Film-Rezensionen.de data . . . 56

4.1.6. Sentiment Analysis task . . . 56

4.1.7. Discussion . . . 59

4.2. Projection . . . 60

4.2.1. Method . . . 60

4.2.2. Evaluation . . . 64

4.2.3. Discussion . . . 66

4.2.4. Summary . . . 67
4.3. Heidelberg Sentiment Treebank & RNTN . . . 68
4.3.1. Hyperparameter selection . . . 69
4.3.2. Impact of training data on performance . . . 70
4.3.3. Enhancing the RNTN with Word Clusters . . . 72
4.3.4. Recovering lost sentiment information . . . 75

5. Combining the Components **82**
5.1. Ensemble Learning . . . 82
5.1.1. Features . . . 82
5.1.2. Learners . . . 83
5.1.3. Initial Results - Learner and Feature Selection . . . 84
5.1.4. Evaluation on Heidelberg Sentiment Treebank . . . 85
5.1.5. Discussion . . . 85
5.2. Uptraining . . . 88
5.2.1. Uptraining and the Heidelberg Sentiment Treebank . . . 89
5.2.2. Clustered Uptraining . . . 91
5.2.3. Clustered Uptraining: Recovering lost sentiment information . . . 92
5.3. Error Analysis . . . 95
5.3.1. Accuracy at n . . . 95
5.3.2. Production-Level Accuracy . . . 99
5.3.3. Contrastive Conjunctions . . . 102

6. Conclusion **105**
6.1. Answers to Central Questions . . . 105
6.1.1. Questions 1, 2 and 3: RNTN for German . . . 105
6.1.2. Question 4: Improving Sentiment Analysis Results . . . 106
6.1.3. Question 5: Most useful methods regarding cost or performance . . 106
6.1.4. Question 6: Relationship between cost and performance . . . 107
6.2. Summary and Perspectives . . . 108
6.2.1. Perspectives . . . 109

A. Reproducing the Results **111**
A.1. Basic Instructions . . . 111
A.2. HeiST data . . . 111

1. Deutsche Zusammenfassung

1.1. Sentiment-Analyse

Die Erkennung von Sentiment befasst sich mit der Untersuchung von Texten auf positive und negative Emotionen oder Stimmungen [Feldman, 2013]. Vor allem die zunehmende Menge von nutzergenerierten Bewertungen von Produkten, Filmen und politischen Themen im Web 2.0 machen die *Sentiment Recognition* interessant. Die Analyse von Sentiment ist auch auf einzelnen Satzteilen möglich. So lassen sich einerseits einzelne Aspekte von Produkten, wie beispielsweise die Bildqualität einer Kamera, erfassen. Andererseits bietet die Analyse von Teilen die Möglichkeit, gemäß des Fregeschen Kompositionalitätsprinzips die Bedeutung des ganzen Satzes anhand seiner Teile und deren Zusammensetzung zu erfassen. Einen neuen Ansatz für diese *Compositional Sentiment Recognition* beschreiben Socher et al. [2013] mit ihrem Recursive Neural Tensor Network, das auf mehrschichtig annotierten Daten die Wortrepräsentation und Kompositionsfunktion lernt. Ältere korpusbasierte Verfahren arbeiten typischerweise auf Dokument- oder Satzebene und arbeiten mit flachen, nichtkompositionellen Dokumentrepräsentationen auf Basis einzelner Worte oder kurzer Phrasen [Wang and Manning, 2012].

Neben den korpusbasierten Methoden sind lexikonbasierte Verfahren zu nennen. Ein Sentiment-Lexikon enthält eine Liste von Wörtern mit der zugehörigen Kategorisierung als *positiv* oder *negativ*. Die Erstellung derartiger Wörterbücher erfolgt entweder aufwändig manuell [Taboada et al., 2011], durch Übersetzung von Lexika anderer Sprachen [Remus et al., 2010] oder ebenfalls korpusbasiert mit statistischen Verfahren [Waltinger, 2010]. Ausgehend von einem Lexikon kann das Sentiment eines Dokuments bestimmt werden, indem jedes Wort des Dokuments nachgeschlagen wird. Dann gewinnt die häufigste Klasse oder, bei gewichteten Lexika, die Klasse mit dem größten summierten Gewicht. Genauere Verfahren berücksichtigen kompositionelle Aspekte durch simple Regeln für Negation (*nicht*) und Intensivierung (*sehr*) oder durch rekursive Anwendung von komplexeren Kompositionsregeln [Taboada et al., 2011].

Meine Masterarbeit befasst sich mit Verfahren für kompositionelle Sentiment-Erkennung im Deutschen.

1.2. Daten

Ich erstelle ein Korpus, in dem sowohl ganze Sätze als auch einzelne Satzteile bis hinab zur Wortebene auf einer Fünf-Punkte-Skala von *sehr negativ* bis *sehr positiv* annotiert sind. Die Annotation erfolgt über einen Crowdsourcing-Dienst, wo Laien die Phrasen in kleinen Einheiten zusammengefasst gegen Bezahlung bearbeiten können.

Entsprechend der binären Kompositionsfunktion des RNTN leiten sich die Satzteile aus den binarisierten Syntax-Bäumen der vollständigen Sätze ab. Das Korpus umfasst 592 Sätze aus Film-Rezensionen, die jeweils das endgültige Urteil des Rezensenten als Fazit zusammenfassen. Die so entstandene **Heidelberg Sentiment Treebank** (HeiST) nutze ich zur Evaluation der vorgestellten Methoden als auch zum Training der überwachten Verfahren.

1.3. Verfahren

1.3.1. Lexikon-Induktion

Ich stelle ein schwach überwachtes (*weakly supervised*) Verfahren vor, um aus benutzergenerierten Rezensionen und den zugehörigen numerischen Bewertung (1-5 Sterne) ein domänenspezifisches Sentiment-Lexikon zu lernen. Die verwendeten Rezensionen stammen aus einer anderen Quelle als HeiST und konzentrieren sich gelegentlich auf Aspekte des Mediums, z.b. Bildqualität, anstatt auf den Film selbst. Hier benutze ich ein überwachtes Verfahren zur Domänenanpassung. Für die Erstellung des Lexikons benutze ich einen Regressionslerner, der auf einer Unigram-Repräsentation der Filmrezensionen eine Funktion lernt, mit der sich die zugehörige Stern-Bewertung vorhersagen lässt. Der Regressionslerner lernt hierbei Gewichte für einzelne Wörter, die positive beziehungsweise negative Einflüsse auf die Bewertung realisieren. Durch die $l1$-Regularisierung des Algorithmus werden bevorzugt Gewichte für besonders informative Wörter gelernt. Der so gelernte Gewichtsvektor lässt sich direkt als Sentiment-Lexikon benutzen.

1.3.2. Projektion

Unter dem Begriff der sprachübergreifenden Sentiment-Analyse sind Verfahren versammelt, die Ressourcen in eine Quellsprache nutzen, um Texte einer Zielsprache zu verarbeiten[Denecke, 2008a]. Insbesondere für die englische Sprache sind umfangreiche Korpora und Werkzeuge vorhanden, die in der Zielsprache nutzbar gemacht werden sollen. Neben spezielleren statistischen Verfahren [Popat et al., 2013] ist maschinelle Übersetzung beliebt. Ein mögliches Verfahren umfasst die Übersetzung der Texte von der Zielsprache in die Quellsprache. Die Texte werden mit existierenden Werkzeugen annotiert und die Ergebnisse auf die ursprünglichen Texte projiziert. Für die hier beschriebene kompositionelle Sentiment-Analyse ergibt sich die Schwierigkeit, die Ergebnisse auf Phrasen unterhalb der Satzebene zu projizieren. Ich beschreibe ein neues Verfahren für sprachübergreifende kompositionelle Sentiment-Analyse, das über die automatische Angleichung der zugrundeliegenden Syntax-Bäume [Tiedemann, 2010, Volk et al., 2010] zwischen Deutsch und English die Sentiment-Labels abbildet.

1.3.3. RNTN und Verbesserungen

Ich trainiere und evaluiere das RNTN [Socher et al., 2013] auf HeiST. Weiterhin entwickele und evaluiere ich zwei Verfahren, welche die Genauigkeit des RNTN auf kleineren Datenmengen verbessern. HeiST ist mit 592 Sätzen relativ klein verglichen mit der Stanford Sentiment Treebank von Socher et al. [2013]. Die Testdaten enthalten also viele Wörter, die dem resultierenden Modell unbekannt sind. Ich generalisiere auf Wort-Ebene, indem ich Wörter durch Wort-Cluster [Brown et al., 1992] ersetze [Popat et al., 2013]. Im zweiten Verfahren unterteile ich die Wort-Cluster anhand eines Sentiment-Lexikons, um den Verlust an Sentiment-Information auf Wortebene durch die Generalisierung zu mildern.

1.3.4. Kombinationen

Die bisher beschriebenen Verfahren haben unterschiedliche Stärken. Ich kombiniere alle Verfahren und das SentiWS-Lexikon [Remus et al., 2010] erfolgreich in einem **Ensemble**-Verfahren [Kittler, 1998, Xia et al., 2011]. Ein Nachteil des Ensemble-Verfahren ist die komplexe Feature-Extraktion. Um eine Vorhersage für einen Satz zu treffen, wird die Ausgabe aller zugrundeliegenden Systeme benötigt. Ein anderes Verfahren umgeht dieses Problem, nutzt aber trotzdem die kombinierten Stärken aller Systeme. Für **Uptraining** [Petrov et al., 2010] wird das RNTN auf einer Kombination aus der manuell annotierten HeiST und maschinell per Ensemble annotierten verrauschteren Daten trainiert. Das Uptraining-Verfahren lässt ebenfalls sich mit den cluster-basierten Verbesserungen kombinieren.

1.4. Ergebnisse

System	Node F1	Root Accuracy
SentiWS	0.624 **	0.711
Regression-Lexikon	0.391 **	0.569 **
Projektion	0.591	0.751 **
RNTN	0.590	0.679
RNTN Cluster	0.591	0.701
RNTN Clusters Unterteilt	0.599 **	0.687 *
Ensemble	0.658 **	0.765 **
Uptraining	0.612 **	0.669
Uptraining Cluster	0.599	0.706
Uptraining Cluster Unterteilt	0.606 **	0.718

Tabelle 1.1.: Ergebnisse auf Satzebene (Root Accuracy) und Phrasenebene (Node F1). */** kennzeichnet statistische Signifikanz $p < 0.05/p < 0.005$.

Auf Satzebene berechne ich *Accuracy*, auf Phrasenebene den F1-Wert. Alle Werte sind jeweils der Mittelwert über die Klassen *positiv*, *negativ* und *neutral*. Zum Vergleich ist das SentiWS-Lexikon [Remus et al., 2010] aufgeführt. Der Test auf statistische Signifikanz wird jeweils im Vergleich zu RNTN mit dem Approximate Randomization Test [Riezler and Maxwell, 2005] durchgeführt. Auf Phrasenebene zeigen Ensemble und Uptraining jeweils eine Verbesserung. Die unterteilten Cluster verbessern die Performance ebenfalls. SentiWS ist trotz seiner Einfachheit signifikant besser als das RNTN. Auf Satzebene ist insbesondere das sprachübergreifende Projektionsverfahren und Ensemble besser als das reguläre RNTN. Die unterteilten Cluster verbessern das RNTN ebenfalls, allerdings nicht im Kontext des Uptraining. Das über den Regressions-Lerner erstellte Lexikon ist den anderen Verfahren in beiden Aufgaben unterlegen. Weitere Auswertung, insbesondere linguistischer Phänomene wie Negation, befinden sich im Hauptteil meiner Masterarbeit in Abschnitt 5.3.

2. Introduction

The increase in recent years in user-contributed content in the Web 2.0 provides ample opportunity to monitor public opinion on a wide variety of topics. People offer their insights on politics, products, companies and news on social media such as Facebook and Twitter. It is useful for marketing and news agencies to monitor and aggregate the opinions expressed by the users. Companies might want to analyze the general sentiment after a product release. This process of **Opinion Mining** is also called **Sentiment Analysis**. Another useful resource of information are product reviews or movie reviews in public fora or on shops like Amazon.com. The vast amount of reviews makes it hard to gather a definite opinion, a general sense of “thumbs up” or “thumbs down”, regarding a product or a movie. Analysis of the sentiment expressed in those reviews can be useful in showing the big picture. More advanced applications include the analysis of individual aspects of products, such as display quality or battery life for a cell phone.

The field of Sentiment Analysis is ripe with many promising, useful approaches which already yield commercial success. Yet many interesting challenges remain.

Among these challenges, one finds the need to acquire lists of sentiment-bearing words which serve as the foundation of many sentiment analysis methods. I present a new way to derive a domain-specific list of sentiment-bearing words for movie reviews.

Furthermore, to obtain a meaningful sentiment analysis for a given document in light of linguistic issues such as negation, one must derive the meaning of the whole from the meaning of its parts and their composition. To this end, I present several methods to extend an existing supervised method of handling compositionality to require significantly less training data.

Finally, most work on Sentiment Analysis has been done on English data. I present a new corpus annotated with sentiment labels on the subsentential level along with a new method of cross-lingual sentiment analysis.

My thesis is structured as follows: I will first introduce the problem in depth in section 2.1. Then follows an overview of related work in the field of sentiment analysis in general and in subfields such as cross-lingual sentiment analysis in section 2.2. In section 2.3, I present a short outline on how I solve the problem. In chapter 3, I show how I acquire the annotated corpus and then present the individual components of the solution in chapter 4. In chapter 5, I show how I combine the components into an improved solution.

2.1. The Problem

Movie reviews are a popular text domain for research on sentiment analysis because annotated data has been available early on [Pang et al., 2002][1] and because sentiment-bearing parts are easily identified. Little research has been done on German sentiment analysis for the movie review domain and I aim to hopefully contribute some insights. A detailed description of previous work on sentiment analysis for German follows in section 2.2.4. Negation (*This movie is not interesting*) is a big challenge for sentiment analysis along with the use of intensifiers (*This movie is very interesting*).

These challenges are not readily handled by traditional bag-of-words machine learning methods without extensive feature engineering. They motivate a compositional approach which not only considers sentiment polarity, but also sentiment strength. Rule-Based methods, on the other hand, handle syntax-based phenomena better, but require extensive rule engineering and sentiment dictionaries with good coverage.

Socher et al. [2013] present a fully supervised approach which recursively learns the compositionality function based on parse trees where each node is annotated with the sentiment label for its phrase. As each node in the parse tree needs to be manually annotated, obtaining training data is expensive.

My thesis is thus concerned with the development and evaluation of a low-cost approach for compositional sentiment analysis for German movie reviews.

In particular, I will attempt to answer the following questions:

1. How can I adapt the supervised Socher et al. [2013] method to German with as little monetary effort as possible?
2. Do I have to create an expensive, manually annotated data set?
3. In particular, how can I use the existing English data by Socher et al. [2013] for German sentiment analysis?
4. Can I increase system performance for a given data set with existing NLP tools and resources, without necessarily annotating more data?
5. Which methods are particularly useful regarding cost and/or performance?
6. Can I quantify the relationship between cost and performance?

2.2. Related Work

2.2.1. Sentiment Analysis

Sentiment Analysis, also known as **Opinion Mining** can be divided into the following sub-tasks [Feldman, 2013]:

[1]The authors maintain an incomplete list of 100 papers using their data at `http://www.cs.cornell.edu/People/pabo/movie-review-data/otherexperiments.html`

- Document-level sentiment analysis
- Sentence-level sentiment analysis
- Aspect-based sentiment analysis
- Comparative sentiment analysis
- Sentiment lexicon acquisition

I will present some related work. More comprehensive surveys of the literature are available in Tang et al. [2009], Kumar and Sebastian [2012]. As more than 1000 relevant papers have been published in the past decade in communities encompassing NLP, data mining, information retrieval and others [Liu, 2012], I focus on a broader overview of some popular techniques.

Document-Level Sentiment Analysis For document-level and sentence-level sentiment analysis, the goal typically is to categorize the text either as positive or negative.

Analyzing Aspects Aspect-based or feature-based sentiment analysis is performed on a more fine-grained level. A document is assumed to contain different opinions on different aspects which are to be extracted individually. Consider product reviews for digital cameras where a reviewer might comment on different aspects such as battery life, picture quality, general handling and so on. More formally, Liu [2012] defines an *opinion* as a quintuple:

$$(e_i, a_{ij}, s_{ijkl}, h_k, t_l)$$

where e_i represents an entity (the camera), a_{ij} an aspect of e_i (the camera display), s_{ijkl} is the sentiment regarding a_{ij} of e_i (good camera display), h_k is the opinion holder and t_l is the time at which the opinion is expressed.

For a typical product review, the author is the opinion holder and the product in question is the opinion target. The time is typically also provided along with the review. For more complex documents, e.g. political news with comments by different politicians, extraction of holder and targets becomes more complicated.

Comparative Opinions Comparative sentiment analysis is used where an user does not provide a direct opinion. Consider a fictional review sentence: *I like the iPhone much better than my old Nokia.* Here, two entities are being compared. The goal for a comparative sentiment analysis is to extract the preferred entity instead of a single polarity (negative, positive) label [Feldman, 2013].

Lexicon Acquisition Sentiment Lexicons contain lists of word with an associated **prior polarity** (also called **semantic orientation** [Turney, 2002]). The polarity can either be ordinal, i.e. *positive* or *negative*, or represented on a continuous scale between the two extremes. These lexicons are either created manually or obtained via statistical methods from the data.

In this work As evident from the categories presented so far, sentiment analysis is a complex field. In this work, I am only concerned with **document-level sentiment analysis** and **sentiment lexicon acquisition**.

Albeit the main focus is on document-level sentiment analysis, the compositional method I use entails successful sentiment analysis below the sentence level, on sentence fragments. This does not mean that aspect-based sentiment analysis is performed, i.e. neither entities nor their aspects are extracted. Comparative sentiment analysis is also not considered. Some instances like *I love this movie more than X* may convey sentence-level polar sentiment as well as comparative sentiment and are thus within the scope of this work[2]. In the quintuple framework, document-level sentiment classification is then defined as follows [Liu, 2012]. Find the sentiment for some aspect GENERAL of some entity:

$$(\_, GENERAL, s, \_, \_)$$

where entity e, opinion holder e and time t are either known or not relevant. In the description of related work that follows, I include text categorization and subjectivity detection where applicable. Sentiment analysis and subjectivity detection both are problems in the field of text categorization. Subjectivity detection is concerned with categorizing text as subjective or objective and is thus intimately related to sentiment analysis. Subjectivity detection can be a part of sentiment analysis. Here, the task consists of the classification of text into one of *positive*, *negative* or *neutral*. As an alternative, subjectivity detection can be used to filter out neutral items which improves sentiment analysis accuracy [Pang and Lee, 2004]. Once neutral items are removed, the classification task becomes binary with classes *positive* and *negative*.

Document-Level and Sentence-Level Sentiment Analysis

For document-level sentiment analysis, the overall sentiment of a document is desired. Generally, the document is authored by a single author and contains an opinion on a single object which is to extracted. In case of sentence-level sentiment analysis, the objective is similar, just on a more fine-grained level.

Two different approaches can be found in the literature: methods based on machine learning and methods based on sentiment dictionaries.

Dictionary-Based Methods A very simple method requires just a list of sentiment-bearing words along with their prior sentiment polarity. These words are then looked up in the document to be analyzed and the most frequent class wins. The dictionaries necessary for this approaches can either be generated manually or obtained by various methods from data.

Turney [2002] presents an unsupervised approach to the classification of product reviews. The reviews are extracted from Epinions.com and already classified as *recom-*

[2]The decision of whether a particular statement conveys traditional polar sentiment lies with external annotators.

mended or *not recommended* by their authors. Based on two seed words for positive and negative word classes, he obtains further sentiment-bearing words from an unannotated corpus. Turney [2002] extracts candidate phrases from a corpus consisting of 410 reviews with Part-of-Speech patterns such as *adjective noun* or *adverb adjective*. The context is required for disambiguation purposes. Turney [2002] cites *unpredictable* as an example, which can be positive in the context of movie reviews (*unpredictable plot*), yet negative in a automotive review (*unpredictable steering*). In a second step, the author computes the **semantic orientation** for a given phrase as the point-wise mutual information (PMI) between the phrase and *excellent* minus the PMI between the phrase and *poor*. The probabilities necessary for the PMI calculation are not estimated from the review corpus, but rather via queries to the Altavista search engine. The overall semantic orientation for a review is then obtained by calculating the average semantic orientation for all phrases in the review. A positive SO yields a *recommended* classification or *not recommended* otherwise. The approach yields an accuracy of 74.39%, compared to a majority class baseline of 59%.

Taboada et al. [2011], in their **SO-CAL** system, use a manually annotated sentiment lexicon along with several rules to compute the final semantic orientation of a sentence. Their lexicon consists of four separate lists for adjectives, nouns, verbs and adverbs. The adjective lexicon is obtained by hand-tagging their Epinion.com review corpus consisting of eight different product categories. For nouns, verbs and adverbs, additional words were added from a 100-text subset of the movie review corpus by [Pang et al., 2002] and from positive and negative word lists obtained from the General Inquirer lexicon. Labels for semantic orientation (SO) ranging from -5 for *extremely negative* to $+5$ for *extremely positive* are assigned to each word by a native English speaker, with neutral words being excluded. The resulting lexicons contain 2252 adjectives, 1142 nouns, 904 verbs and 745 adverbs. From the adjective lexicon, the SO value of 483 commonly occurring words is evaluated using the Mechanical Turk crowdsourcing platform.

Two tasks are presented to the workers. The first one requires a word to be categorized into positive, negative and neutral. If the current lexicon is valid, then the authors expect more non-neutral items with lower SO values being classified as neutral than items with higher SO values. In other words, terms with stronger sentiment should be classified as neutral less often.

The second task requires the worker to decide whether two polar adjectives are equally strong or if one is stronger than the other. The expectation here is that two words which are closer on the SO scale should be rated equally strong more often than words which have more SO distance. The crowdsourced evaluation confirms the expectations and the dictionaries can be considered valid.

The inclusion of verbs and nouns presents an improvement over Turney [2002] as these word classes can contain sentiment-bearing words. Taboada et al. [2011] show the following example to motivate the inclusion of the additional word classes:

- The young man strolled+ purposefully+ through his neighborhood+.
- The teenage male strutted- cockily- through his turf-.

Another novelty is the addition of a intensifier and negation lexicon. Intensifiers can increase (*very good*) or decrease (*slightly useful*) the SO of a sentiment-bearing word. Taboada et al. [2011] choose to use relative increases (decreases) by assigning percentages to intensifiers instead of using fixed offsets to add to (subtract from) the SO of a word. The authors give *somewhat sleazy* as an example, where *sleazy* has an SO value of -3 and *somewhat* has a modifier percentage of -30%. The resulting SO value is calculated as $-3 \times (100\% - 30\%) = 2.1$. The relative approach has the added advantage of adding a higher SO value to already strong phrases, e.g. *really great* will see a bigger absolute increase than *really OK*.

To handle negation, the authors favour **shift negation** over **flip negation**. For flip negation, a negated word simply has the sign of its SO value inverted. In case of *not sleazy*, the associated SO would flip from -3 to 3. However, *not sleazy* is still not a very positive expression and the negation implies more of a shift toward the positive than a complete reversal of the polarity. This insight is reflected in the shift negation approach where the SO value of a negated word is shifted by 4. In the example given, *not sleazy* would be given the SO value -1 which seems more appropriate. It is important to determine the scope of negation to determine for which terms the polarity needs to be shifted. Taboada et al. [2011] present two approaches to detect negation scope. Both involve looking backwards from a sentiment-bearing phrase to find a negator term. The first option searches backwards until a clause boundary marker such as punctuation or connectives (e.g. *but*) is found. The second, more conservative, approach searches backwards as long as the words are found in a part-of-speech-specific skip list. The authors use the second approach throughout their paper as it works better. Another linguistic phenomenon that changes sentiment value is the **irrealis** which indicates non-factual contexts. Consider the following two examples (Taboada et al. [2011]):

- I thought this movie would be as good as the Grinch, but unfortunately, it wasn't.
- But for adults, this movie could be one of the best of the holiday season.

In the first example, the irrealis marker *would* reverses the polarity of *good*. In the second example however, *could* does not influence the sentiment orientation of *best*. The authors identify several irrealis markers:

- Modal verbs
- Conditional markers (like *if*)
- Some intensional verbs (like *expect*, *doubt*)
- Questions
- Words enclosed in quotes

The authors handle these cases by ignoring any SO-bearing words in the same clause as an irrealis marker. There is one exception for cases like rhetorical questions:

- ...he can get away with marketing this amateurish crap and still stay on the bestseller list?

Based on the definite article inside the NP of the sentiment-bearing adjective, the irrealis blocking is ignored in this case.

The authors further argue that humans favor positive language, thus resulting in a positive bias for sentiment analysis systems. To counteract this problem, the SO for any negative expression is increased by 50% to give more cognitive weight to the negative class. Repeated usage of the same sentiment-bearing word suggests lack of substance and thus, the nth appearance of the same expression is reduced to $\frac{1}{n}$ of the original SO value.

In their evaluation, Taboada et al. [2011] compare the performance of their system across different lexicons and across several rule sets. For the lexicons, they find that accuracy increases significantly by 4.81 percentage points when adding all word classes (78.74%) compared to using only adjectives (73.93%). For the comparison of the rule sets, the authors find a significant increase of 12.7 percentage points between using no rules at all (66.04%) and using a rule set consisting of shift negation, intensification, irrealis blocking for modal verbs, negation weighting and repetition weighting (78.74%).

As part of the data comes from the popular Pang et al. [2002] data set, the results obtained by Taboada et al. [2011] are comparable. The machine-learning approach by Pang et al. [2002] yields 87.15% overall accuracy, compared to 78.74% obtained by Taboada et al. [2011]. However, machine-learning methods often fit the domain on which they are trained too well and performance drops considerably in cross-domain settings. Taboada et al. [2011] show that their SO-CAL system performs well across different types of reviews as well as different corpora from news, blog and social media domains. Thus, cross-domain performance is a strong point of SO-CAL.

The authors also evaluate other dictionaries using the SO-CAL rules. They find that their manually annotated dictionary performs best. Among the compared lexicons is a PMI-based dictionary similar to Turney [2002] where the term probabilities are estimated with Google instead of Altavista. The accuracy with this dictionary is significantly lower at 62.98% compared to 74.78% for SO-CAL. The authors attribute the success of their dictionary to their strict rules regarding the exclusion of ambiguous words and the inclusion of fewer words rather than more words. The validity of their dictionary generation approach is also attested by the superior performance in the mechanical turk task described above. Nevertheless, for all dictionaries, the application of the SO-CAL rules improves the performance compared to simply computing the average SO over all terms in a review.

Machine-Learning Based Methods Unlike dictionary-based methods, machine learning methods offer strong performance without extensive rule engineering. Pang et al. [2002] compare three machine learning methods and several document representations. They perform their experiments on a movie review corpus extracted from IMDB.com which is still widely used today. Document labels are assigned based on the star ratings given by the authors of the reviews. The reviews are then represented as unigrams or bigrams. Basic negation handling is realized by prefixing every unigram between a negation and the next punctuation mark with *NOT*. The individual features are represented as either binary vector entries, indicating the presence of a word, or as feature frequency.

The authors report 82.9% classification accuracy for the SVM classifier with the unigram feature set and binary feature representation. In particular, the frequency-based feature representation performs significantly worse. The combined unigram and bigram feature set also performs strongly at 82.7%, while bigrams only is significantly worse at 77.4% for the Maximum Entropy classifier. The Naive Bayes classifier performs worst except for the unigram features represented with their frequency counts where it yields 78.7%. Two extended unigram feature sets are presented. The authors add the part-of-speech tag to an unigram to serve as a form of word-sense disambiguation, yet obtain one percentage point less than without POS tags at 81.9%. The authors also add position information to indicate whether an unigram occurs in the first quarter, fourth quarter or middle half of a document. This heuristic aims to replicate the fact that movie reviews are typically divided in parts such as a plot summary and an overall opinion. This setup yields 81.6% accuracy.

Another experiment where the set of unigrams is restricted to adjectives only results in 77.7% classification accuracy. This comparatively bad result further indices that adjectives alone, while informative, are not enough to accurately perform sentiment classification. This finding is consistent with the results of Taboada et al. [2011] as discussed above.

In summary, the authors obtain the best classification accuracy at 82.9% with a basic unigram model combined with basic negation handling and a standard SVM classsifier. The impact of the negation handling is not quantified, but the authors state that there is a "negligible, but on average slightly harmful, effect on performance" if the negation markers are removed.

In comparison with standard text categorization tasks, the authors find sentiment analysis to be a harder problem as sentiment is often not as overtly realized as topics. In particular, they cite what they deem the *thwarted expectations* phenomenon, where a review author will list negative (or positive) expectations about a movie, then come to a conclusion bearing the opposite sentiment:

- "This film should be brilliant. It sounds like a great plot, [...] However, it can't hold up."
- "I hate the Spice Girls. [...] The plot is such a mess, it's terrible. But I loved it."

As a closely related case, they cite the *good actor trapped in a bad movie* phenomenon where positive descriptions of an actor contrast with negative opinions about the movie itself.

In more recent work, Wang and Manning [2012] present guidance for the selection of the correct machine learning approach. Multinomial Bayes (MNB) outperforms SVM for snippet tasks where each document consists of only 20 words. For longer document such as complete movie reviews (over 200 words), the SVM variants win. Unlike Pang et al. [2002], Wang and Manning [2012] report significant improvements using bigram features. The authors also propose a novel SVM variant with NB log-count ratios as features which is a robust performer across all text lengths.

Combinations of Lexicon-Based and ML-Based Methods Choi and Cardie [2008] present a method to combine lexicon-based and machine-learning based methods. For the traditional lexicon-based methods, they consider simple heuristics like voting where the most frequent label wins, some voting-like strategies with enhanced negation treatment and two compositional strategies. The latter consist of simple syntax-based rules and binary composition functions.

Consider the phrase *[destroyed]* _ *VP [the terrorism]_NP* [Choi and Cardie, 2008]. The rule **Polarity([VP]_NP)** fires and invokes the composition function **CompoMC([VP], [NP])**. Since the VP *destroy* is a content-word negator, the composition rule flips the polarity of the negative second argument, *the terrorism.* The resulting polarity is positive. In case of conflicting polarities between first and second argument, the majority class (**CompoMC**) wins. An alternative strategy, **CompoPR** is available, which gives priority to the first argument. For complex arguments, the Polarity and Compose functions are called recursively.

In their basic classification experiment, they encode several feature sets for each expression. The lexical features consist of the words in the expression themselves and, separately, the lemma for each word. Dictionary features describe word categories in an attempt to abstract away from individual word forms. The voting-based features encode the results of a subset of the heuristics described above. These features form a simple, flat bag-of-words representation.

In their **Classification with Compositional Inference** (CCI) approach, the authors train a classifier to provide the input for the composition function described above. In particular, the classifier replaces the syntax-based rules such as *Polarity([VP]_[NP])* and directly outputs *{positive, negative, negator, none}* as an *intermediate decision variable* for each word. In this classification scheme, the composition function *Compose* is used in the loss function to penalize incorrect decisions. An initial version of the intermediate decision variables is constructed from the polarity lexicon and updated during training for incorrect predictions.

Choi and Cardie [2008] evaluate their approaches on the Multi-Perspective Question Answering corpus (MPQA) [Wiebe et al., 2005], [Wilson et al., 2005]. This newswire corpus is annotated with sentiment labels on phrase level, not on sentence-level or document-level as described before. In particular, the boundaries for sentiment-bearing expressing are given and do not need to be extracted. This likely results in a simpler task. The simple voting heuristic yields 86.5% accuracy. Methods considering only function-word negators perform at around 82%. The inclusion of content-word negators increases the accuracy by 5 points to 87.7%. The authors argue that content-word negators have been underrepresented in previous research where they were mostly only available as negative entries in a sentiment lexicon without any special treatment. The rule-based composition methods yield 89.7% for *CompoMC* and 89.4% for *CompoPR.* The learning-based methods using bag-of-words features performs at 89.1% and thus beat the simple heuristics, but not the rule-based compositionality. The CCI learning algorithm obtains 90.7% accuracy and is the strongest performer. The results show that the integration of content-word negators can provide strong results even for simple heuristics. The application of compositionality rules further improves performance. The combination of learning, compositionality rules and sentiment lexicon provides the best performance.

Deep Learning for Compositional Sentiment Analysis The previous sections have shown that sentiment analysis benefits from a compositional approach. Intensifiers, shifters and sentiment-bearing words with different polarities and strength make for a challenging task.

Socher et al. [2013] present a supervised approach to learn the compositionality function for sentiment analysis. A sentiment treebank consisting of 11,855 sentences is constructed via crowdsourcing. Every node of the binarized parse trees is manually annotated with a sentiment label. This results in 215,154 labeled phrases.

The authors propose a novel model called the **Recursive Neural Tensor Network** (RNTN). At prediction time, an RNTN processes a parse tree bottom-up. The word vectors for the leaves are learned during training and thus known at prediction time. The tree is binary and every node has one parent and exactly one sibling. The vector for a parent node is computed with the same tensor-based composition function from its two child nodes [Rudolph and Giesbrecht, 2010]. The sentiment polarity for the node, or rather its underlying phrase, is then obtained from a separate classifier.

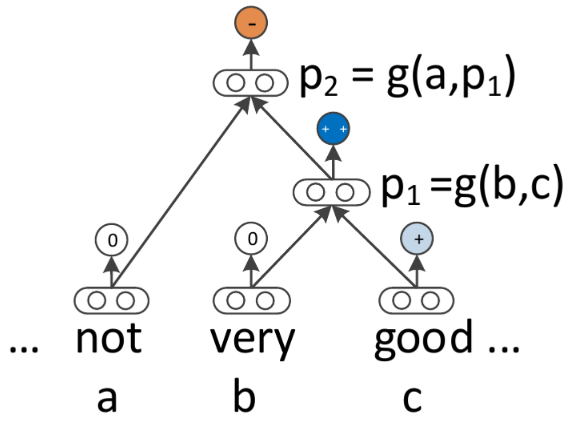


Figure 2.1.: Recursive Neural Networks for Sentiment (Socher et al. [2013], Fig. 4)

The process is illustrated in figure 2.1. The vectors for intermediate nodes such as *very good* are constructed bottom-up by application of the composition function g: $p1 = g(very, good)$. Note that the composition function is always the same. The sentiment label for this hidden vector is then predicted with a softmax classifier. The RNTN is of particular interest here as I will be working closely with the model and the required sentiment treebanks in my thesis.

The Stanford Sentiment is derived from the rottentomatoes.com movie review excerpt dataset originally published by Pang and Lee [2005]. The excerpts do not constitute full reviews. Instead, an excerpt typically is a single sentence which captures the sentiment of the author towards the movie.

After parsing the sentences with the Stanford PCFG Parser [Klein and Manning, 2003], the subtree spans are considered as phrases which are to be annotated. 215, 154 phrases are uploaded to Amazon's crowdsourcing platform, Mechanical Turk. In the annotation task, workers are presented with a phrase accompanied by a slider with 25 different values. The slider has tickets at intermediate values for *very negative, negative, somewhat negative, neutral* and corresponding entries for positive sentiment. The authors find that even a five-class scheme captures the main variability in the manual annotation. The resulting labels are *very negative, negative, neutral, positive, very positive.* In contrast to typical binary classification tasks, they call this task **fine-grained sentiment classification**. The authors find that sentiment distribution varies between phrases of different lengths. For unigrams, the majority of the phrases is neutral. At length 20, only 20% of the phrases remain neutral. For full sentences, the label distribution is more uniform, with less neutral sentences than sentences of either positive or negative sentiment.

The authors report 85.4% accuracy on the binary sentence classification task and 80.7% accuracy on the fine-grained all-node classification task. This results represents state of the art for binary sentiment analysis on the Pang and Lee [2005] data set. The fine-grained accuracy at the sentence level is lower at 45.7%. This is easily explained as the classification task with five classes is much harder than with two classes. The RNTN beats all baselines, including other recursive models and bag-of-words style ML approaches. Closer inspection of the results reveals that the RNTN performs strong on shorter n-grams in comparison to ML methods. The effects of compositionality are stronger on shorter phrases while bag-of-words models need more tokens to perform well.

The authors further test the compositional aspects of the model with regards to negation. In the **contrastive conjunction** test, they extract sentences with a structure like "X but Y" and evaluate the accuracy on this subset. Consider an example sentence:

> "There are slow and repetitive parts, but it has just enough spice to keep it interesting."

Two subphrases with contrasting sentiment labels are linked with "but" and the model should be able to accurately classify both subphrases. This setting includes neutral phrases. The accuracy obtained here on 131 sentences is 44% for the RNTN, which represents an increase of 4 points compared to earlier work (37%).

In the **high level negation** task, an artificial set of negated sentences is created from existing data. For the subtask **negating positive sentences**, the data consist of positive sentences and the corresponding negated version. To show that the model learns more than a simple negation rule, e.g. a polarity flip if negation is detected, the authors create a second subtask, **negating negative sentences**. Here, already negative sentences are negated further. The goal for the model is to increase positive activation. As I have mentioned in previous discussions of negation, negating a negative phrase does not necessarily result in a positive phrase, but in a less negative phrase.

For **negating positive sentences**, the RNTN beats the second best model by 19 percentage points at 71.4% accuracy. In the **negating negative sentences**, the RNTN shows an increase in positive activations in 81.4% of all cases, whereas the second best performer yields 54.6%. These results show that the RNTN can accurately model complex negation cases, even for more subtle phrases such as *the most compelling variations* versus *the least compelling variations*.

Sentiment Lexicon Acquisition

Basic sentiment lexicons consist of word lists annotated with the prior polarity label. Optionally, an indicator indicating the strength of the sentiment might be given. Beyond lists of sentiment-bearing words, word lists of polarity shifters and negators are also possible. From these word lists, several methods are available to obtain the final sentiment value from the prior sentiment polarity. Methods include simple voting-based mechanisms [Turney, 2002], application of rule cascades [Taboada et al., 2011] or machine learning [Choi and Cardie, 2008]. As the acquisition of domain-specific or domain-independent lexicons is clearly important to performance of sentiment analysis, I will present several approaches.

Manual Annotation Many sentiment lexicons are simply obtained by means of manual annotation. [Pang et al., 2002] create a weak baseline by collecting a few sentiment-bearing words from students. Other work create proper sentiment lexicons which are available for re-use. Taboada et al. [2011] manually create a lexicon of sentiment-bearing words with the strength rated on a five-point scale for each polarity with additional lexicons for intensification and negation. Wiebe et al. [2005] present the MPQA (Multi-Perspective Question Answering) corpus where they annotate 10, 657 sentences from 535 news documents. The annotation goes beyond the sentence level to the aspect level. The sentiment source, the polarity, the strength and the sentiment (opinion) target are annotated.

Corpus-Based Automatic Extraction Turney [2002] use seed words (*excellent*, *poor*) and a PMI-based algorithm to find sentiment-bearing words along with a score indicating sentiment strength. Esuli and Sebastiani [2006] describe their SentiWordNet. They use a semi-supervised method to label each WordNet synset (and thus, the associated terms) with three scores, one for each of *positive*, *negative* and *neutral*. Like Turney [2002], the authors start from seed sets consisting of positive and negative word. These sets form the training set and are iteratively expanded by following lexical (*also-see*) and antonymy relations in WordNet. The expanded sets along with a set of neutral terms are used to train a set of eight ternary classifiers. The terms are represented in a standard VSM by their WordNet glosses normalized by TF-IDF. For a given term, the ratios between positive, negative and neutral labels given by the predictions of the committee of learners form the distribution of the sentiment labels. The sum of the score is 1. The resulting **SentiWordNet** has 24.63% synsets which are not neutral. The distribution of the subjectivity scores suggest that only few synsets are truly polar, i.e. 10.45% of synsets have objective scores less than 0.5. Strongly subjective synsets where the objective score is less than 0.125 make up only 0.56%. A total of 115, 000 synsets is annotated.

Mohammad et al. [2009] describe their approach to generate a high-coverage semantic orientation lexicon from a thesaurus. They use the Macquarie thesaurus and call their method **Macquarie Semantic Orientation Lexicon** (MSOL). They use a set of affix patterns (affix seed list, ASL) to look up a list of positive and negative seed words. For example, if a word pair *honest* - *dishonest* exists in the thesaurus, the pattern for the *dis-* affix fires. The affixed words are generally considered negative. The thesaurus contains $1,000$ categories of, on average, 120 closely related words. Within these categories, terms are further grouped into sets of near-synonymous words called **paragraphs**. The paragraphs are then searched for the seed terms. If a paragraph contains more positive than negative seed terms (and vice-versa), then all words in the paragraph are considered positive (negative). A word can appear in multiple paragraphs, in which case the most common label wins. The authors find that their MSOL approach has a better coverage of the GI lexicon (74.3%) compared to SentiWordNet (60.3%). Other automatic approaches still yield superior coverage (83.3%). In a phrase polarity identification task on the MPQA corpus [Wiebe et al., 2005, Wilson et al., 2005], the MSOL(ASL) performs best across all phrases as measured by balanced F-Score ($F1 = 0.539$). In particular, the set of seed words without thesaurus-based expansion performs only at $F1 = 0.242$ which proves the usefulness of the expansion process. If GI terms are used as seeds instead of ASL, then MSOL(GI) improves upon MSOL(ASL) at $F1 = 0.615$. Merged seed sets (MSOL(GI+ASL)) performs best at $F1 = 0.617$. The authors show that their simpler thesaurus-based method of sentiment lexicon acquisition outperforms the WordNet-based SentiWordNet.

Star Ratings A part of my thesis is concerned with the creation of a sentiment lexicon from weakly labeled movie review. The weak labels are the star ratings which the reviewers assign to the movie. In this section, I will review some related work on the usage of star ratings in prediction and sentiment analysis tasks.

Maas et al. [2011] present a combination of an unsupervised probabilistic model of semantic similarity and a distantly supervised logistic regression learner of semantic orientation. Word representations for semantic similarity are learned in an unsupervised manner similar to LDA on a movie review corpus. The semantic orientation of words is learned by mapping the star ratings onto $[0; 1]$ and then training a logistic regression predictor. The final objective that is optimized in the combined learning process is the sum of both objectives for semantic similarity and semantic orientation.

The method is evaluated on the [Pang and Lee, 2004] data set. The learned word representations are used to obtain a document representation from a bag-of-words vector of a document. A standard SVM classifier is then trained on the data for a binary classification task. The semantic component separately obtains 87.10% accuracy. The combined method yields 84.65% accuracy. In a semi-supervised setting where additional unlabeled data is used to train the semantic component, the performance increases to 87.05%. The semi-supervised combined representation concatenated with the original bag-of-words vectors gives best accuracy at 88.90%. However, the appraisal groups by Whitelaw et al. [2005] reach 90% on the same data.

Rill et al. [2012] describe a method to extract polarity-bearing phrases from German Amazon movie review titles and star ratings. Simple patterns are used to extract candidate phrases, e.g. any single noun with an exclamation mark (*Madness!*). For adjectives, review titles consisting of a single adjective or adverb+adjective are considered among other patterns. Review titles are rejected if they are formulated as questions, if they contain irrealis markers (*could*, *should*), markers of irony such as emoticons and quotation marks, or if they could be ambiguous.

For each candidate phrase, the **opinion value** (OV) is calculated from the average star rating of the reviews titles containing that phrase. A three-star rating is considered neutral and the values are mapped to a continuous scale from -1 to 1 accordingly.

A further correction of the opinion value is necessary due to the skewed distribution of the star ratings. More than half of the reviews have five-star ratings, leading to a J-shaped distribution with the minimum at around 2 stars. This leads to a bias of neutral phrases towards positive opinion values. The authors thus reclassify phrases following the J-shaped distribution as neutral. Opinion values for non-neutral phrases are weighted by the relative frequencies of star ratings of the whole sample.

The end result is a list of 3.210 phrases and words. 700 of these are neutral ($-0.33 < OV < 0.33$), 770 weak subjective ($0.33 < OV < 0.67$) and $1,740$ strong subjective ($OV > 0.67$ resp. $OV < -0.67$). The authors include multi-word phrases to capture negators, intensifiers and shifters. This alleviates the need for compositionality rules as multi-word expressions are already covered by the lexicon.

The authors provide no evaluation besides the observation that existing polarity lexicons like GermanPolarityClues [Waltinger, 2010] provide twice as many entries. Unlike GermanPolarityClues, the new resource provides sentiment strength on a continuous scale rather than binary assessments.

The authors further argue that the generated lexicon likely is domain-specific and thus better suited for review-based applications than other text domains.

Nguyen et al. [2014] introduce a **rating-based feature** for sentiment classification. The authors train a unigram-based regression model (SVM) on an external data set of movie reviews titles along with their review scores. Only movie reviews with scores other than 5 or 6 stars, i.e. non-neutral reviews, are used.

This model is then used to predict a review score for reviews from both train and test sets. The predicted score is used alongside standard unigram, bigram and trigram features to train a standard SVM classifier for positive and negative movie reviews.

The authors evaluate their method on the Pang and Lee [2004] data set. Using only the rating-based feature, the accuracy is 88.2%. Adding the n-gram features boosts performance to 91.6% which represents state of the art on this data set.

It is also possible to predict the star ratings using the review text. Pang and Lee [2005] present a metric labeling algorithm which exploits the varying degrees of similarity between the class labels, i.e. one- and two-star reviews are more closely related one- and five-star reviews.

This short overview on usage of star rating has hopefully shown that user-assigned scores are a useful feature in sentiment analysis tasks. Many review data sets already derive the semantic polarity labels from user-supplied star ratings [Waltinger, 2010, Ghorbel and Jacot, 2011, Pang et al., 2002, Pang and Lee, 2005]

The overview presented here has shown that star ratings are not only useful as full supervision for document labels [Bikel and Sorensen, 2007], but also for distant supervision as weak labels for dictionary acquisition and sentiment analysis.

2.2.2. Cross-Lingual Sentiment Analysis

Many approaches to tackle the problem of sentiment analysis extend naturally to other languages. Supervised classification methods are typically easily applied to new languages. All that is required is an annotated corpus to train the classifier. Lexicon-based methods can also be applied to other languages. Simple methods, which sum over the occurrences of positive and negative tokens, only require a sentiment lexicon. More sophisticated methods which use patterns to capture negation and scope effects require more engineering efforts.

Anotated corpora are not always available for resource-scarce languages. The creation of new corpora for a new language requires significant labeling effort. This is not always possible because of approaching deadlines or financial considerations. Thus, several methods exist to leverage resources in a resource-rich language for sentiment analysis in a resource-scarce language.

Machine Translation

Banea et al. [2008] describe several experiments where machine translation is used to generate tools and resources from English (source language) to Spanish and Romanian (target languages) for subjectivity detection.

Experiments In their **first** experiment, the MPQA corpus is automatically translated to the target language and the sentence-level sentiment labels are projected to the translation. The result is an annotated corpus in the target language.

The **second** experiment assumes that no annotated corpus is available in the source language. Instead, a subjectivity classifier is used to annotate raw text in the source language. The annotated text is then translated to the target language and the labels are projected. The raw data is taken from a partial Romanian translation of the SemCor corpus. A subset of this corpus manually is annotated with sentiment labels and serves as a gold standard.

The **third** experiment is similar to the second experiment with a key difference: the raw text exists in the target language and is translated to the source language. There, the text is annotated automatically and the labels are projected back to the source language.

The intermediate result in all three experiments is an annotated corpus in the target language. SVM and NB classifiers are trained on this corpus and tested on manually annotated data in the target language.

In a **fourth** experiment, the data generated in the third experiment is evaluated. In this setting, the quality of the generated resource is of interest. The first three experiments consider the performance of a classifier trained on the generated resources. The fourth experiment serves as an upper bound as no losses are introduced by machine learning.

Experiments one and two are repeated for Spanish.

Discussion The SVM always beats the NB classifier.

For Romanian, the second experiment performs best at 69.44% accuracy The first experiment, where the English MPQA is translated, yields 66.07% accuracy. The authors propose as an explanation the more subtle nature of subjectivity in MPQA. Subjectivity is not necessarily overtly expressed, but rather as implied cues which are easily lost in translation. For the second and third experiments, the OpinionFinder tool is used to create the training data. OpinionFinder relies on a subjectivity lexicon, i.e. on overt mentions of words. These surface features are more resilient to the effects of translations.

The upper bound found in the fourth experiment is 71.83%. The SVM classifier introduces losses of 2.39 points.

In earlier work, Mihalcea et al. [2007] perform experiments similar to #2 and #3. Instead of using a machine translation system, they use the manually translated SemCor corpus directly. The source (English) side is annotated with OpinionFinder and the labels are translated to the target side (Romanian). A Naive Bayes classifier is then trained on the data. This setup allows an estimate of the influence of machine translation versus a “perfect” manual translation. Banea et al. [2008] state that using a parallel corpus instead of an automatically translated corpus can yield improvements by about 4 points.

For Spanish, overall results are better than for Romanian. The MT system for English-Spanish is better than for English-Romanian. The MPQA-based first experiment outperforms the second experiment. The authors attribute this to the improved MT which more accurately translates MPQA.

The authors describe an interesting additional experiment. One might generally assume that using cross-lingual sentiment analysis is an inherently lossy process. That is, information will be lost in translation and performance on the target side will never be as good as on the source side.

They annotate the English side of their parallel SemCor corpus with OpinionFinder. The labels are projected to the Romanian side of the parallel corpus. Separate classifiers are trained on both data sets and tested on the test portions of the parallel SemCor. In this experiment, all settings are equal except for the language. One would expect the English setting to perform much better as OpinionFinder is developed for English. However, Romanian yields 69.84% accuracy while English yields 60.32%.

The authors attribute this surprising result to the fact that Romanian is a highly inflected language which provides additional subjectivity markers. In other words, subjectivity classification for Romanian is inherently easier than for English.

Although a system specifically developed for Romanian is likely to provide even better performance, it is my interpretation of both experiment four and the last experiment presented that “easier” source languages can offset some of the penalties incurred by machine translation.

Other Work The work described so far only considers the translation of annotated corpora. Mihalcea et al. [2007] provide an additional experiment where they translate the subjectivity lexicon used by OpinionFinder into the target language using standard bilingual dictionaries. As the original OpinionFinder requires a parser to bootstrap additional extraction pattern, a modified OpinionFinder and the translated lexicon are used to process the Romanian data. In comparison to the projection method described above, performance is approximately 30 points worse as measured by the F measure.

Denecke [2008b] also uses machine translation to build a cross-lingual sentiment analysis system. German movie reviews are translated into English and classified with three different systems. The single rule-based system counts occurences of terms found in SentiWordNet [Esuli and Sebastiani, 2006]. Two supervised machine learning classifiers are presented as well. The first system represents documents as character 8-grams. The second system is a meta-classifier which uses the sums of SentiWordNet terms for positive, negative and objective terms The supervised systems are trained on an IMDB movie review corpus. Out of the three classifiers, the meta-classifier performs best at 66% accuracy for the binary sentiment classification problem (negative, positive).

Corpus-Based Methods

Machine Translation is an obvious way to automatically transfer textual resources from source to target language. As Popat et al. [2013] note, sentiment analysis is a less resource-intensive task then machine translation. Thus, it is hard to justify the reliance on a machine translation system for cross-lingual sentiment analysis. In fact, not all languages are covered by publicly available machine translation systems.

Popat et al. [2013] investigate word clusters as a way to solve the sparsity problem in sentiment analysis. They consider syntagmatic clusters [Brown et al., 1992] and paradigmatic clusters (WordNet, Christiane [2012]).

The authors also investigate the usefulness of multilingual WordNet instances for cross-lingual sentiment analysis. Additionally, they propose two methods for cross-lingual cluster linking which do not require expensive resources such as WordNet.

Methods In their first approach, they represent each sentiment-labeled document in the source language as a set of WordNet sense identifiers. Using a MultiDict (a multilingual WordNet), the documents are then represented as the set of WordNet sense identifiers in the target language.

The sentiment classifier is trained on the projected data. The test data in the target language is also represented as a set of WordNet synset identifiers.

This approach corresponds to the first experiment described in Banea et al. [2008]. The main difference is that the corpus is that a dictionary is used to translated on the level of synsets identifiers instead of using a standard MT system.

For the syntagmatic Brown clusters, Popat et al. [2013] describe two cluster linking methods which both require a parallel corpus.

The first method, **direct cluster linking**, assumes that the documents for both languages have been clustered independently. The clusters are then linked across languages by considering the scores of the word alignments. A source side cluster is linked to the target side cluster which maximizes the sum of the alignment scores between words in source and target side clusters.

The second method, **cross-lingual cluster**, does not perform clustering and linking in two separate steps. Instead, cross-lingual clustering maximizes the likelihood of monolingual word-cluster pairs for both source and target as well as the alignments between source and target. Traditional monolingual clustering only maximizes the joint likelihood of words and clusters in a single language.

As a baseline, the authors employ a traditional MT-based CLSA method.

Results Popat et al. [2013] evaluate the cross-lingual methods on Hindi and Marathi, two widely spoken Indian languages. For Marathi, no MT system is (publicly) available, making it a good candidate for the methods presented. The MT-based baseline yields 63.13% accuracy for Hindi. The WordNet-based projection and direct cluster linking reach 53.80% and 51.51%, respectively. Cross-lingual clustering outperforms all other methods at 66.16%. For Marathi, the WordNet-based projection performs worse (54.00%) than direct clustering (56.00%). Cross-Lingual clustering also beats all others at 60.30%.

From the results presented by Popat et al. [2013], it is clear that cross-lingual sentiment analysis does not necessarily require complex machine translation systems.

Criticism on CLSA

Machine Translation Balamurali et al. [2013] argue that sentiment analysis is best done by creating resources native to a language instead of relying on cross-lingual techniques. In particular, they claim that annotating documents with sentiment labels to create native resources is easy and affordable. With minimal effort, the results will beat cross-lingual methods. In their own words:

> "If you want to do sentiment analysis in your language and have a limited amount of money, spend the money in creating polarity marked documents for your language, instead of using MT and then doing CLSA."

To verify their hypothesis, they compare four cross-lingual methods against a standard SVM classifier with unigram feature for four languages. They find that the SVM clearly outperforms all cross-language systems. Furthermore, they find that 500 annotated documents are enough to saturate the SVM classifier for all languages. Adding more documents does not increase accuracy further. The annotation process for 500 documents only requires about ten hours. From these findings, the authors conclude that the creation of native resources is more viable than CLSA.

I offer my thoughts on these findings in the context of my cross-lingual sentiment analysis method in section 4.2.

Domain Adaption Duh et al. [2011] also consider the role of machine translation in cross-lingual sentiment classification. In their experiments, they find that MT is indeed ripe for CLSA as very few label mismatches are produced. In other words, the MT system rarely mistranslates a word such that the semantic orientation of the word flips.

What they observe, however, is that there may be a problem of instance mismatches. The output of the MT may be valid; that is, the semantics of the translation match the original and the sentence is grammatically correct and fluent. However, the distribution of words in the MT output may be different from naturally occurring English. As an example, the MT output might use *excellent* more often than (the somewhat colloquial) *awesome*. A classifier trained on naturally produced language might not have seen *excellent* as often, thus producing poorer results.

This problem ought to be solved with standard domain adaptation methods. The authors observe that these methods do not work quite as well for language adaptation (on translated data) as for domain adaption (e.g. movie VS book reviews). From this observation, they conjecture that adapting from artificially generated MT text requires different, new methods.

As a possible limitation of this study, I'd like to note that the reviews are translated word-by-word. As the MT system does not see the full sentence, instance mismatch problems might be compounded.

2.2.3. Ensemble Learning

Ensemble learning improves classification accuracy by combining features and classifiers of different types [Ho et al., 1994].

Ensemble learning typically fits into one of two categories, according to Kittler [1998]. The first category uses **distinct representations** for each classifier. Here, different learners are trained on different representations of the same data. The second category uses **identical representations**. Typically, either different learning algorithms are trained on the same data set or the data is divided into subsets and a single learner is trained on each subset.

Regardless of distinct or identical representations, the base classifiers can be combined in several ways. Broadly speaking, two combination methods are available. The first one is based on hard-coded **rules**. A prime example is the majority voting rule. Here, the class for an item is predicted by a set of classifiers. The class which is predicted most often among the set of classifiers is then selected as the predicted class. The second method involves the estimation of classifier weights on training data. This is essentially a **supervised** classification setting where the predictions made by the base classifiers form the feature space. Ho et al. [1994] evaluate several combination functions on an optical character recognition task and report best performance for supervised methods which are fitted to the data.

In theory, different combination strategies must be used depending on the underlying classifiers and their output. For example, for distinct representations, Kittler [1998] assumes that the representations are independent of each other. This enables use of the product rule to combine the a posteriori probabilities given by the base classifiers.

In practice, these distinctions and the underlying assumptions cease to matter when the combination functions are learned from the data. Kittler [1998] states:

> "When linear or nonlinear combination functions are acquired by means of training, there is very little distinction between the two basic scenarios. [...] This probably explains the successes achieved with heuristic combination schemes derived without any serious concerns about their theoretical legitimacy."

Ensemble Learning for Text Categorization

Dong and Han [2004] compare several ensemble methods in a text categorization task. They distinguish between homogeneous and heterogeneous ensemble methods. Homogeneous methods use the same base learner which are trained on different subsets of the data. Heterogeneous methods, on the other hand, combine different base learning algorithms. The authors evaluate several homogeneous ensemble methods on the Reuters 21578 data set. The ten most frequent topics are chosen as classes. For the base learners, they choose SVM and a Naive Bayes variant. In the heterogeneous setting, they combine both base learners with a neural network as the combination function. They find that homogeneous ensemble methods improve NB considerably. SVM, on the other hand, performs worse when used in an ensemble setting. The heterogeneous classifier can beat other classifiers in three out of ten classes. Overall, the single SVM classifier obtains best performance. Thus, the ensemble methods do not an advantage in the experiments presented by the authors.

Ensemble Learning for Sentiment Analysis

Li et al. [2007] report on their efforts to combine classifiers for a binary sentiment categorization task on movie reviews. Unlike Dong and Han [2004], their base classifiers are not obtained by subsampling the data or by choosing different machine learning algorithms. Instead, they choose to train the SVM algorithm on distinct representations, i.e. different feature sets, of their data. The feature sets are based on a standard bag-of-words model of the reviews where the binary features indicate presence, not the word frequency. The biggest feature set consists of all unigrams. The POS based feature sets are then subsets of the unigram feature set filtered by a given POS tag or combinations thereof. The classifiers are sorted according to performance. Several rule-based combination methods described by Kittler [1998] are then used to combine the N-best classifiers.

The best performance is obtained a N=3 with unigram, adjective+adverb and adjective feature sets. Combined with the sum rule, 83% precision is obtained. This is an improvement of 2.54 points over the best base classifier (unigrams).

Whitehead and Yaeger [2010] evaluate several ensemble methods in a sentiment analysis task. The data comes from different review domains. The authors use a standard SVM as their base learner. The data is partitioned according to several algorithms and the SVM is trained on the partitions to create the base classifiers. The SVM also serves as a baseline and typically produces accuracies around 85% in the binary classification task. The ensemble algorithms typically increase accuracy by around 1 to 3 points, depending on the algorithm and the data set. The bagging subspace algorithm emerges as a clear winner.

Xia et al. [2011] provide a comprehensive study of different ensemble learning settings for the sentiment classification task. Their feature sets are derived from standard a bag-of-words model, partitioned by parts of speech. Additionally, word relation (WR) features such as higher-order n-grams, syntactic relations between words and the full set of unigrams are used.

Additionally, they evaluate several different classification algorithms as base learners. The authors employ both fixed composition functions and trained methods.

In this experiment, the base learners are created either with different learning algorithms or with different feature sets. The different combinations are evaluated on five widely used data sets.

In their evaluation, the authors find that trained methods yield better results than the fixed rules. The combination of both different feature sets along with different base learners in a single ensemble setting provides the best performance. Crucially, the ensemble settings always perform better than a single concatenation of the input vectors in case of the different feature sets.

Resource-Scarce Ensemble

Xia et al. [2013] present an interesting use of ensemble techniques in a domain adaptation task. In their setting, only a small in-domain training set is available. However, a large out-of-domain training set is available along with a small labeled in-domain data set. The goal in domain adaptation here is to use the large out-of-domain data set to get good classification performance on the in-domain data.

The task can be decomposed into two problems. The first is instance adaptation, where the distribution of vocabulary and word frequency is adapted. Xia et al. [2013] perform instance adaptation by sample selection. They use principal component analysis to obtain the latent concepts, and more importantly, the latent concept distribution, of both in-domain and out-of-domain data. Out-of-domain instances which are close to in-domain instances in the latent concept space are then used as training data.

The second sub-problem is labeling adaptation. Xia et al. [2013] handle the problem of labeling adaptation by applying ensemble learning techniques. Between domains, one feature that might be positive in-domain could be negative out-of-domain. The authors observe that features with different POS tags have different, distinct distributions between domains. Nouns tend to be domain-dependent while adjectives are mainly domain dependent. Based on this observation, Xia et al. [2013], use their feature ensemble method to learn the weights for the different POS classes. Four feature sets based on different POS tags are defined and used to train four Naive Bayes base classifiers on the out-of-domain data. The output of the base classifiers forms the vector space for a perceptron. The perceptron is then trained on the in-domain data.

Classifier performance improves considerably compared to baseline.

2.2.4. German Sentiment Analysis - State of the Art

A multitude of sentiment analysis methods for the English language exist. Most methods extend naturally to other languages, such as German. As English is much more prevalent, there is less literature on Sentiment Analysis for German. In particular, the ubiquituous movie review dataset by [Pang et al., 2002] and its variants is widely used and facilitates easy comparison between different methods. Unfortunately, no such widely used data set is used in the German sentiment analysis community. This makes it hard to find the definite state of the art for sentiment analysis on the German language. However, papers describing new sentiment analysis systems often compare their proposed methods to existing approaches. The authors typically compare their system on the same data. From these papers, some insight in the state of the art of German sentiment analysis can be gained.

Corpus-Based Methods for German Sentiment Analysis

Atalla et al. [2011] introduce a German news article corpus for subjectivity detection. They implement a number of state-of-the-art subjectivity detection methods based on supervised machine learning and evaluate them on their corpus. Although subjectivity detection is not the same as polarity classification, the problem is similar (and relevant!) enough to warrant inclusion. Typically, methods which perform well on subjectivity detection also perform well for polarity classification. Both tasks are part of the sentiment analysis problems. Additionally, they evaluate on the English MPQA corpus [Wiebe et al., 2005] to find out which methods and features are especially useful for German.

Feature Sets The first approach they evaluate is **unique words**. Infrequent words are used to separate the subjective and objective classes. A word is defined as infrequent if it is not one of the 600.000 most frequent words as measured by the BNC (English) and the Leipzig Corpora Collection (LCC).

The second approach is based on **POS trigrams**. The feature space is not based on word forms, but on the POS tags of the words. A sequence of three POS tags forms a single feature.

The third approach is a standard supervised classification approach based on **unigrams, bigrams, trigrams and POS tags**. The feature space consists of word forms, two- and three word sequences and POS tags. A sentence is represented as a vector consisting of all these features.

The **minimum-cut classifier** is the fourth method. This graph-based approach incorporates neighboring sentences in its decision. The underlying idea is that subjective sentences tend to be surround be other subjective sentences.

The fifth method uses only **long-distance bigrams** [Chen et al., 2007]. Also known as gappy bigrams, these features are designed to better capture word history. A 2-distance bigram will skip one word. A sentence of length n is then modeled as $w_1w_3, w_2w_4, \ldots, w_{n-2}w_n$.

The sixth and last method is a standard, **machine translation** based cross-lingual approach to subjectivity detection. The MPQA corpus is automatically translated to

German and used as training data.

All methods are tested with SVM and Naive Bayes learners. These learners are also trained on a standard unigram feature set to form two baselines.

Results The **unique words** feature set is not useful, neither standalone nor when its feature vector is concatenated with the unigram vectors.

The **POS trigram** features also do not beat the baseline. Comparing the results between languages, German benefits more from POS trigrams than English. When adding the POS trigram features to the unigram baseline, performance also does not improve except for NB. However, the NB baseline performs quite poorly compared to the SVM. This indicates that the baseline features already cover much of the contribution by POS trigram.

The long feature vectors in the **unigrams, bigrams, trigrams and POS tags** experiment show a clear improvement for both SVM baselines. Accuracy improves by 1.8 points for German and 1.5 points for English. Naive Bayes does not show any improvement.

The **minimum-cut classifier** uses either Naive Bayes or SVM with the unigram feature set as its base classifier. MinCut with NB does not show any improvement. MinCut with SVM improves, on average, by 1.85 points for German and 1.64 points for English.

Both the MinCut method and the big feature set improve more for German than for English. This may indicate a better fit of the respective methods for German. The authors remark that the baseline for English is considerably better than for German. Thus, it is harder to gain improvements of the same magnitude for English.

The **long-distance bigrams** method performs consistently worse than the baselines. Only when long-distance bigrams are added on top of the baseline unigram feature, performance increases slightly for NB. SVM does not perform better. On average, the long-distance bigrams hurt performance more for English than for German when NB is used. SVM does not show any difference across corpora here.

Machine translation is used to translate the MPQA corpus into English. Several of the methods presented are trained on the translated data and evaluated on the German corpus. The methods are also trained on the German corpus in a cross-validation setting to obtain an upper bound on performance. For the lower upper bounds, the classifiers trained on the translated data perform quite well. The difference often only consists of 1 to 3 points in accuracy. For the best performers (as determined by upper bound), the difference is much larger however. The best classifiers trained on the German data reach approximately 70% while the best cross-lingual classifiers only reach 63.5%.

In summary, the authors find that the presented approaches perform better on English than on German. If an approach improves performance on English, then it also improves on German. The standard unigram feature set performs well for both languages. Both the minimum cut approach and the extended feature set consisting of higher-order n-grams and POS tags improve performance. German benefits more from POS-trigrams and from the extended feature set. According to the authors, this suggests that subjectivity in German depends more on grammatical structures. English subjectivity, on the other hand, is more based on lexical features.

Lexicon-Based Methods for German Sentiment Analysis

Scholz and Conrad [2013] present a new method for opinion mining in newspaper articles. This work mostly highlights and compares lexicon-based methods. This complements Atalla et al. [2011] who focused mostly on supervised machine-learning approaches using standard vector model representations.

Scholz and Conrad [2013] consider **statements** which consist of up to four sentences with the same semantic orientation. The authors test their approach on an annotated German newspaper corpus.

The approach creates a graph from a set of statements. Each word is modeled as a node. Only nouns, verbs, adverbs, adjectives and negation particles are considered. The POS tag is marked in the node. Edges e_{ij} are created between words which co-occur in the same sentence. Associated with each edge is a triple of weights. For each class (negative, neutral, positive), the number of co-occurrences for two words in sentences of the given class is counted.

Once the graph has been created by counting co-occurrence frequencies on the corpus, it can be used to predict the semantic orientation for an unseen statement. Given a unseen statement s, the subgraph G_{sl} for the l-th sentence in s is extracted. Unknown words not seen during training, i.e. those not in the learned graph, are discarded. The subgraph G_{sl} is then used to generate features for a maximum entropy classifier. For a given node, the probability for the node v_i being positive is calculated as the ratio of the sum of all positive appearances to the sum of both positive and negative appearances:

$$P(pos|v_i) = \frac{\sum_{e_{ij} \in G_{sl}} y_{ij}^{pos}}{\sum_{e_{ij} \in G_{sl}} y_{ij}^{pos} y_{ij}^{neg}}$$

Probabilities for negative, subjective and neutral classes are estimated in a similar manner.

For each of the four word classes considered, two features are described. The polarity feature describes the difference between $P(pos|v_i)$ and $P(neg|v_i)$. The subjectivity feature is the difference between $P(subj|v_i)$ and $P(neu|v_i)$.

The final eight feature values are averaged over all sentences of a statement. A SVM is used to obtain the final classification for unseen statements.

Scholz and Conrad [2013] use several existing lexicon-based methods as baselines. As these methods are evaluated on English in the original literature, the authors use the German SentiWS [Remus et al., 2010] sentiment lexicon. Additionally, they extract a sentiment lexicon from their graph which is evaluated separately. Words which appear more frequently in neutral statements are assigned to the neutral class. For the remaining words, a score is calculated as the frequency of occurrence in positive statements minus frequency of occurrence in negative statements divided by appearance in all statements. The word class for pos/neg and $subj/neut$ is assigned based on fixed thresholds.

The baseline methods require lists of intensifiers, negations, polarity shifters etc. Where applicable, these are translated for each system by a domain expert.

The lexicon-based baseline methods consist of SO-CAL [Taboada et al., 2011], Opinion Observer [Ding et al., 2008] and Wilson et al. [2009]. Additionally, RSUMM as a language-independent method is investigated [Sarvabhotla et al., 2011].

SO-CAL and Opinion Observer do not employ supervised learning. Scholz and Conrad [2013] report on additional experiments where an SVM serves as a meta-classifier. The statements are then classified based on the scores of these systems.

SO-CAL The SO-CAL system [Taboada et al., 2011] is described in section 2.2.1.

Opinion Observer Opinion Observer [Ding et al., 2008] is a system for aspect-based sentiment analysis. For product reviews, individual product features and the sentiment associated with them is analyzed. The system uses a sentiment lexicon induced on WordNet [Hu and Liu, 2004] and a list of hand-annotated idioms. Sentiment analysis is performed with a POS-based rule set. The basic idea is identify sentiment words surrounding feature words. The sentiment words are weighted according to token distance in the sentence to the feature word. Negation handling is based on patterns matching both negation particles such as *not* and verb-based negations such as *stop VB-ing*, e.g. *stop doing*.

The context polarity of context-dependent polarity words (as identified by a separate word list) is identified using conjunction rules. Consider the ambiguous *long* (*long battery life* versus *long wait time*). Conjunctions such as *and* are used to discover whether the ambiguous word is typically used in positive or negative contexts (*great display and long life*). If an ambiguous word is successfully disambiguated, its synonyms (antonyms) inherit the same (flipped) polarity.

Scholz and Conrad [2013] adapt the Opinion Observer system to their setting. Instead of classifying product features, whole statements are used. A subset of the original rules is used.

Wilson et al Wilson et al. [2009] is a supervised machine learning approach. Instead of a standard bag-of-words approach, they engineer several feature sets. On top of word token features, the authors add a feature indicating prior word polarity based on a sentiment lexicon compiled from several sources. Two binary features are introduced to handle both local negation in a four word window and subject negation. Shifters are also considered within a four word context window and represented as three binary features for general, positive and negative shifters. The set of polarity modification features requires a dependency parse tree. If a parent (head) modifies a child and the parent is in the sentiment lexicon, the *modifies polarity* feature is set to the semantic orientation of the parent. The inverse relationship *modified by* is extracted if a child is in the sentiment lexicon. If the word is not found in the lexicon, the respective feature is set to neutral. As a special case, the *conj polarity* feature models the semantic orientation of a word's sibling in a conjunction.

Given this feature set, it is clear that the approach proposed by Wilson et al. [2009] uses more linguistic knowledge than a simple bag-of-words model. The authors find that a boosting classifier performs best. A two-step approach where instances are first classified into polar and neutral and only polar instances are handed to the polarity classifier improves performance further. The authors achieve 74.5% accuracy on a four-class task (positive, negative, neutral, both) on an polarity-annotated version of MPQA [Wiebe et al., 2005, Wilson et al., 2005].

RSUMM [Sarvabhotla et al., 2011] consider the reliance of many sentiment analysis approaches on lexical resources a possible problem for resource-starved language. Their RSUMM approach does not require linguistic resources besides sentence splitters and tokenizers. These tools are more easily obtained than complete sentiment lexicons or parsers. As their approach is supervised, they require a labeled corpus for training.

The basis for their method is standard vector space model. They define two metrics to estimate the subjectivity of individual sentences. The average document frequency (ADF) is the average of the document frequency of all terms. A vector *adf* is derived which contains only terms with a frequency greater than the average document frequency.

The second metric, the average subjective measure, counts term occurrences in subjective and objective sentences as the ratio of subjective occurences to the sum of objective and total occurrences. Similar to *adf*, the vector *asm* is contains all terms with a frequency greater than the average subjective measure.

From these vectors, a subjectivity score for each sentence is derived from the sum of vector similarity between the sentence vector and *adf* and *asm*, respectively. The top x subjectivity scores are retained and the sentences are used as a subjective excerpt of the containing document.

In a second step, the subjective excerpts are modeled as n-gram models. The most important n-grams are selected with standard statistical feature selection methods.

These n-grams are then added to a standard unigram (or bigram) representation of the documents and improve classification accuracy greatly. It is also shown that the RSUMM method of selecting subjective sentences as excerpts outperforms a Naive Bayes classifier.

Results Scholz and Conrad [2013] evaluate all systems on a German and on an English data set. I mainly report the findings on the German data. The graph-based approach by [Scholz and Conrad, 2013] performs best at 63.45% accuracy on the three-class problem (negative, neutral, positive). The second best system is RSUMM at 48.3%. The second place for a system which uses comparatively less linguistic features may indicate a problem with the cross-language application of the more sophisticated systems. However, RSUMM is also the second best system on the English data set. This indicates that extensive linguistic processing does not necessarily improve performance in the sentiment analysis task.

The method proposed by Wilson et al. [2009], comes in third. Opinion Observer and SO-CAL get fourth and fifth place, respectively. The graph-based lexicon makes a positive difference for Wilson et al. [2009] (1.75 points) and SO-CAL (4.27 points). For

Opinion Observer, SentiWS performs better than the graph-based lexicon (6.62 points).

In the meta-classifier setting, where the scores for SO-CAL or Opinion Observer serve as input for an SVM, the rankings are different. For Opinion Observer, performance drops drastically with the SentiWS setting and stays the same for the graph-based lexicon. SO-CAL benefits greatly in both settings. In fact, SO-CAL with the graph-based lexicon almost ties with the Wilson method.

It is not surprising to see that the best methods suffer most. Better performing systems, after all, offer more opportunities for degraded performance. The least linguistically complex system suffers most. From these data, there is no clear evidence that a certain system is particularly ill-suited for cross-language adaptation.

2.2.5. Summary

From the various methods and feature sets investigated by Atalla et al. [2011], a SVM classifier with unigram features provides reasonable performance. A better feature set which improves performance considerably adds bigrams, trigrams and POS tags on top of the unigrams. The best method is the MinCut algorithm introduced by Atalla et al. [2011] themselves.

Out of the lexicon-based methods investigated by Scholz and Conrad [2013], their own graph-based approach performs best. The method proposed by Wilson et al. [2009] uses a multitude of linguistically motivated features and achieves third place. The corpus-based RSUMM [Sarvabhotla et al., 2011] does require neither deep linguistic features nor a sentiment lexicon, yet achieves second place.

2.3. My Approach

So far, I have established that a compositional supervised approach can be very beneficial for sentiment analysis, both from existing work [Choi and Cardie, 2008] and from the problems arising from linguistic phenomena.

I detail the steps undertaken to acquire the Heidelberg Sentiment Treebank (**HeiST**) for compositional sentiment analysis in chapter 3.

In chapter 4, I describe and evaluate the methods I use in my thesis for sentiment analysis. The first method is a novel, distantly supervised approach for the extraction of a domain-specific sentiment polarity lexicon from weakly labeled data.

As a second method, I describe a novel tree-based approach for cross-lingual subsentential sentiment analysis [Banea et al., 2008].

As a third method, I use the RNTN [Socher et al., 2013] and HeiST to train a sentiment analysis model. I then present a novel extension of the RNTN based on word clusters and sentiment lexicons.

In chapter 5, I describe methods to combine the previous approaches to obtain superior performance.

In the **Ensemble** method, I combine the previously described methods in an ensemble classification framework.

A second combination method uses the idea of **Uptraining**. Here, I train the RNTN on a combination of HeiST and the output of the Ensemble method. Note that all data and program code required to run the experiments is available to allow indepedent verification and reproduction of my results. See A.

3. Resources

3.1. The Heidelberg Sentiment Treebank

I require annotated data both for training of supervised classifiers and for evaluation purposes. As movie reviews are a common domain and the SST[1] also uses movie review data, I opt to use German movie reviews.

Several professional and hobbyist sites provide movie reviews. I choose Film-Rezensionen.de[2] as their reviews are provided under a Creative Commons license which allows non-commercial use provided that attribution and a similar license for derived work is maintained (CC BY-NC-SA[3]). This license allows me to distribute the annotated data. Unlike Amazon or IMDb, Film-Rezensionen.de is a hobbyist website maintained by a private person[4]. The team page[5] lists five editors and thirteen contributors, ensuring some diversity among reviews and movie taste.

I download 1437 movie review pages using a Python script. I extract the actual review from the HTML using the BeautifulSoup 4.0 HTML parser[6] along with meta data such as author, review title, review categories, movie rating and review date. The movie rating is typically found at the end of a review, indicated by "Wertung: 3 von 5" (*Rating: 3 out of 5*). I am able to extract the movie ratings based on a regular expression patterns for 631 out of 1437 movies. The scores are normalized to the $[0; 1]$ range. The amount of positive reviews dominates the negative reviews (figure 3.1).

The reviews are split into 38661 sentences using the German Punkt sentence tokenizer shipped with NLTK 2.0.4. [Wagner, 2010]. The data set consists of 828819 words and 5698072 characters[7] Movie titles are occasionally marked with <**em**> and <**strong**> tags. These are preserved to recover the occurence of titles later on.

The movie reviews generally contain an objective section describing the plot and a subjective section giving the reviewer's sentiment or a summary thereof. For the purpose of sentiment analysis, I only extract subjective sections. Subjective summaries are often initiated with "Fazit:" (*In summary:*). I use this pattern to extract the following paragraphs up to the end of a review.

Of all 1437 movie reviews, 402 reviews contain a "Fazit:" paragraph. The resulting data set consists of 1184 sentences.

[1]Stanford Sentiment Treebank

[2]http://www.film-rezensionen.de/

[3]https://creativecommons.org/licenses/by-nc-sa/3.0/deed.de

[4]http://www.film-rezensionen.de/kontakt/impressum/ - accessed 27.08.2014

[5]http://www.film-rezensionen.de/kontakt/about-us/ - accessed 27.08.2014

[6]http://www.crummy.com/software/BeautifulSoup/

[7]`wc`, GNU coreutils

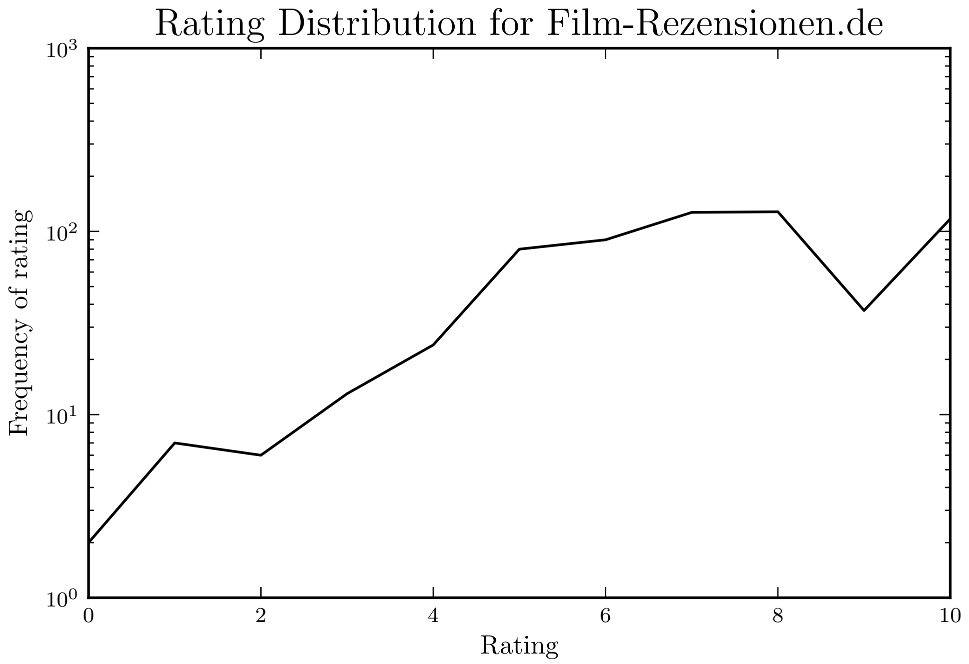


Figure 3.1.: Rating distribution for 631 Film-Rezensionen.de reviews. Y axis is log scaled.

For initial development purposes, I sample 20 sentences starting with “Fazit:”. I parse these sentences using a Berkeley parser trained on the SPRML2013 version of the Tiger corpus with a decision tree for morphological word classes[8]. The parse trees are binarized and unary nodes are collapsed using the Stanford CoreNLP toolkit [Manning et al., 2014]. I extract the segments and annotate them manually in a spreadsheet on a five-point sentiment scale ranging from very negative to very positive. This results in 69 negative (7.7%), 676 neutral (75%) and 151 positive nodes (16.9%). These initial results roughly reflect the results by [Socher et al., 2013], who find that neutral nodes dominate the distribution. Positive nodes occur twice as often as negative nodes. This reflects the property of human language to realize positive sentiment more overtly than negative sentiment.

To make sure my understanding of the annotation task is similar to what Socher et al. [2013] intended and to avoid bias, I manually re-annotate the first 10 sentences of the SST development set. For the coarse-grained sentiment analysis task, i.e. differentiating between negative, neutral and positive, I obtain an accuracy of 80.5%. This is sufficient for initial experiments. For more reliable results, I turn to crowdsourcing to obtain more data from unbiased annotators.

[8]Parser provided by Dr. Yannick Versley.

3.1.1. Annotation with Crowdflower

Crowdsourcing is a low-cost way to label large amounts of data. Workers from various geographic locations can perform small tasks for small amounts of money. I model my crowdsourcing task mainly based on Socher et al. [2013] with some key differences. I use the CrowdFlower[9] service instead of Amazon Mechanical Turk. CrowdFlower can distribute tasks to several channels where different workers work on the tasks. One of these channels is Mechanical Turk, so CrowdFlower can be seen as a frontend to mechanical turk.

Socher et al. [2013] have shown that a five-point scale is sufficient. For this reason, I opt for a five-point radio button system instead of the 25-position slider originally used by Socher et al. [2013]. The instructions provided to workers are a simplified, translated version of the original instructions. In particular, I omit some technical details. I add an instruction stating that phrases are not necessarily grammatically well-formed, in which case workers should use their best judgment. Please see figure 3.2 for a screenshot of the annotation interface. The figure shows the preview provided by CrowdFlower and does not necessarily match the interface used by a channel partner.

Data preparation

Out of the 1182 “Fazit:” sentences, I randomly sample 591 sentences. I parse these sentences with the German PCFG parser included in the Stanford Parser package (version 2013-11-12) [Rafferty and Manning, 2008]. The German parser is trained on the Negra corpus which uses a shallow annotation scheme for PP nodes. In the shallow scheme, the constituents making up a NP are direct children of the PP and not attached to a NP. In the deep scheme, the NP is annotated as such. The headfinder rules shipped with the tree binarizer in CoreNLP require a deep annotation scheme to fire properly. I implement the PP tree deepening rules described by Samuelsson and Volk [2004] in a Java program based on the Stanford CoreNLP API [Manning et al., 2014] to insert appropriate NP nodes. See 3.3 for an example of a problematic PP and the result of rule application.

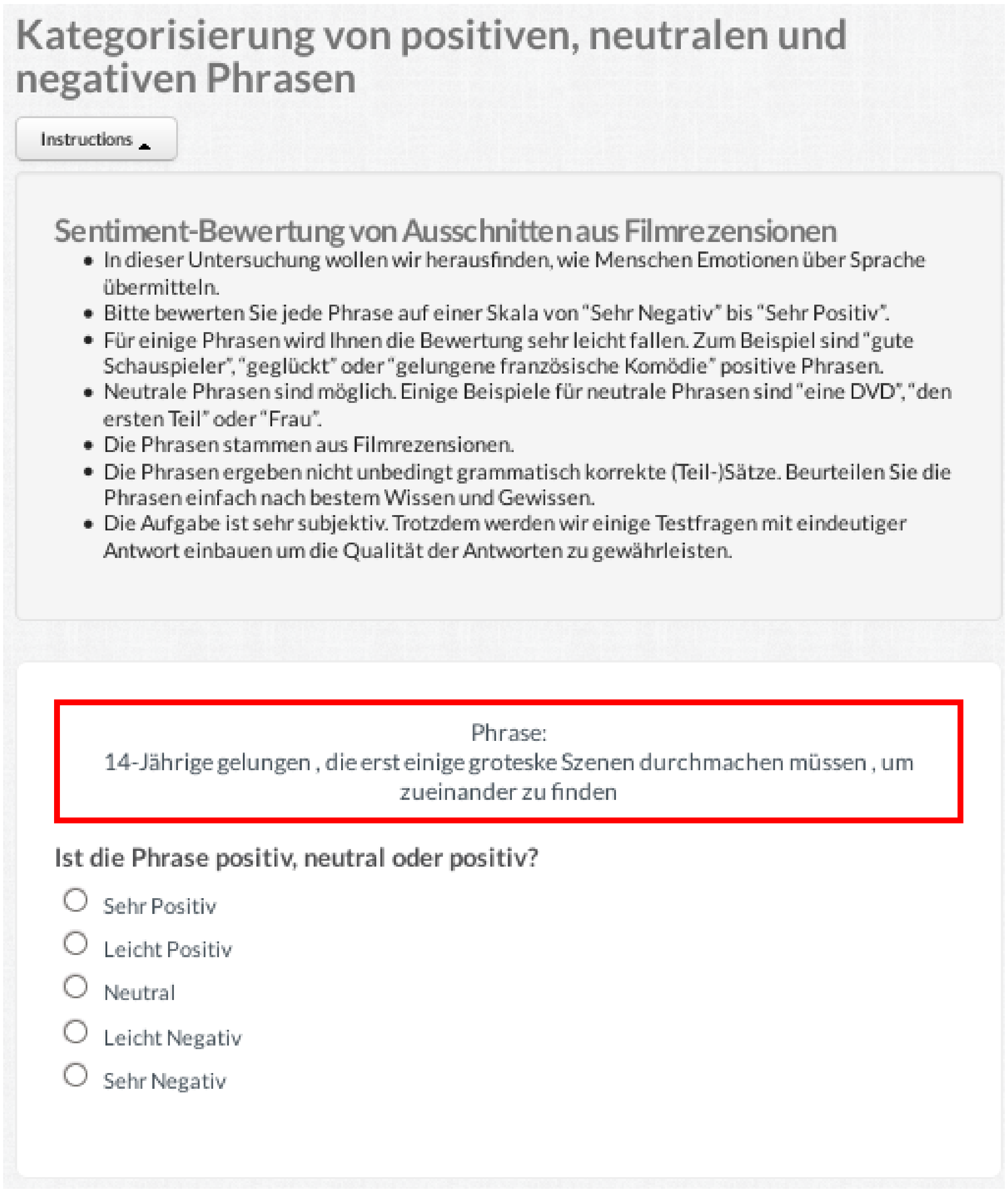
Note that Samuelsson and Volk [2004] describe several rules for unary node deepening which I do not require for the task at hand.

The subtree span associated with each node is extracted. Duplicate spans are removed and the spans for the first 10% of the sentences are annotated via CrowdFlower.

High-Precision Subjectivity Classification

I use the initial set from the first 20% of the sentences to build a high-precision subjectivity classifier. The goal is to safely minimize the amount of neutral phrases I upload to CrowdFlower while making as few false negative errors as possible. A false negative error here is an error where a subjective (positive/negative) phrase is classified as objective. Any errors in this filtering step will reduce the success of supervised methods depending on the data.

[9]http://www.crowdflower.com/


Kategorisierung von positiven, neutralen und negativen Phrasen

Sentiment-Bewertung von Ausschnitten aus Filmrezensionen

- In dieser Untersuchung wollen wir herausfinden, wie Menschen Emotionen über Sprache übermitteln.
- Bitte bewerten Sie jede Phrase auf einer Skala von "Sehr Negativ" bis "Sehr Positiv".
- Für einige Phrasen wird Ihnen die Bewertung sehr leicht fallen. Zum Beispiel sind "gute Schauspieler", "geglückt" oder "gelungene französische Komödie" positive Phrasen.
- Neutrale Phrasen sind möglich. Einige Beispiele für neutrale Phrasen sind "eine DVD", "den ersten Teil" oder "Frau".
- Die Phrasen stammen aus Filmrezensionen.
- Die Phrasen ergeben nicht unbedingt grammatisch korrekte (Teil-)Sätze. Beurteilen Sie die Phrasen einfach nach bestem Wissen und Gewissen.
- Die Aufgabe ist sehr subjektiv. Trotzdem werden wir einige Testfragen mit eindeutiger Antwort einbauen um die Qualität der Antworten zu gewährleisten.

Phrase:
14-Jährige gelungen , die erst einige groteske Szenen durchmachen müssen , um zueinander zu finden

Ist die Phrase positiv, neutral oder positiv?

- ○ Sehr Positiv
- ○ Leicht Positiv
- ○ Neutral
- ○ Leicht Negativ
- ○ Sehr Negativ

Figure 3.2.: Screenshot of the CrowdFlower task preview.

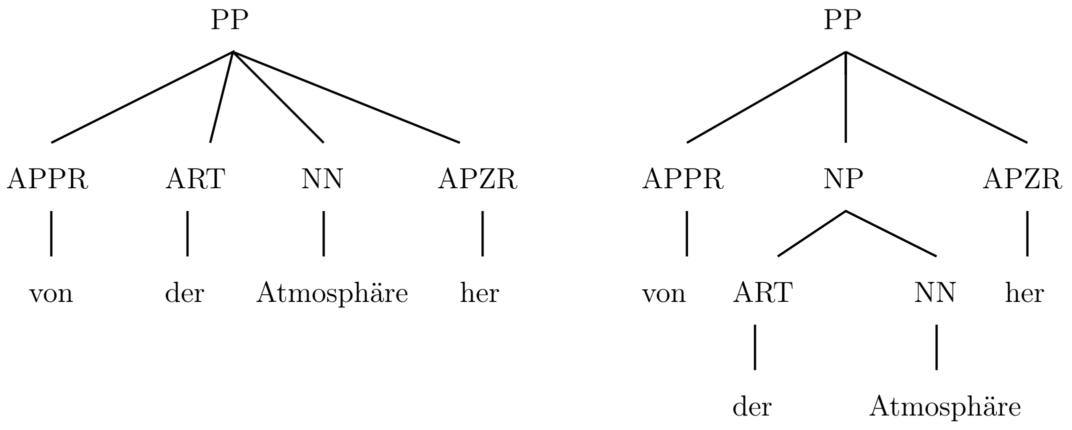


Figure 3.3.: Example of default and deepened PP.

Phrases are represented as a combination of several features. One feature set is derived from the SentiWS dictionary [Remus et al., 2010] (version 1.8c). Phrases are split into individual tokens with the NLTK TreebankWordTokenizer. Some special cases which are not tokenized properly like "ich.und" are handled using regular expressions ("ich", "und") and any punctuation is removed. Lower-cased tokens are then looked up in the positive and negative SentiWS word lists. Positive and negative sentiment orientation values are summed and added as separate features. SentiWS provides not only lemmas, but also word forms, which are also considered for lookup.

Another feature is based on the part-of-speech (POS) tags of the phrase. The POS tags for a phrase are obtained with the Stanford POS tagger version 3.0 [Toutanova et al., 2003] and the German model. Every POS tag is a separate feature and the phrase is represented by the frequency of the POS tags.

The last feature is a variation of the rating-based feature introduced by Nguyen et al. [2014]. I train a Lasso regression learner on German movie reviews downloaded from Amazon.de and their accompanying star ratings. The model is then used to predict a star rating for a phrase. The predicted rating is used as real-valued feature. The process is described in more detail in section 4.1

I use scikit-learn [Pedregosa et al., 2011] to build the classifier. The scikit-learn toolkit offers several machine-learning algorithms and allows simple evaluation of different choices. I evaluate random forest, adaptive boosting, gradient boosting, SVC with a linear kernel and the extra-trees learner. I also compare different combinations of the feature sets. In table 3.1, the introduction of the POS feature increases precision to 92.4% while hurting recall slightly. This is acceptable as I prefer precision over recall for this task.

As 92.4% is still too noisy, I only consider the most confident predictions made by the classifiers. The classifiers output class probabilities for each test sample. The most likely class is then assigned as the predicted class. The most confident predictions are those where the difference between class probabilities is maximal. The test set is sorted by this confidence measure (i.e. the difference in class probabilities) and the top 50% (20%) are

Features Precision	Precision	Recall	Acc
SentiWS	0.883	0.953	0.868
SentiWS + Regression	0.885	0.957	0.874
SentiWS + Regression + POS	0.924	0.936	0.894

Table 3.1.: Subjectivity detection with GradientBoostingClassifier.

Features	Precision	Recall	Acc
SentiWS + Regression @ 50%	0.940	0.994	0.947
SentiWS + Regression @ 20%	0.964	0.981	0.961
SentiWS + Regression + POS @ 50%	0.984	0.992	0.980
SentiWS + Regression + POS @ 20%	0.988	0.987	0.977

Table 3.2.: Most confident predictions for subjectivity detection with GradientBoosting-Classifier.

used as final test set. The results are shown in table 3.2. The precision is considerably higher for this subset, reaching up 98.8%. Please note that the recall measure is not based on the full test set, but rather on the test samples giving the most confident predictions. As I am essentially removing 50% (80%) of the test set, the recall measured across the full test set is likely much lower. When the subjectivity classifier is used as a filtering step for the CrowdFlower annotation task, phrases not matching the confidence cut-off are simply considered subjective (or unknown) and uploaded to CrowdFlower for annotation.

Table 3.3 presents an evaluation of different machine learning algorithms. Performance does not differ much between different classifiers. I use GradientBoostingClassifier as it consistently performs well. I also test combinations of different feature sets and different classifiers. There is still little variation in performance between classifiers so the details are omitted for brevity.

Based on initial experiments on 20-sentences, I use the SentiWS + Regression + POS features with the GradientBoostingClassifier and retain the top 50% confident predictions.

The classifier is then trained on the first set of results from the CrowdFlower annotation task and used to filter objective phrases. Inspection of the filtered phrases reveals that some phrases I consider subjective are wrongly removed. For example, “eine der spektakulärsten Szenen dieses Jahres” (*one of the most spectacular scenes of this year*) or “schrecklich enttäuscht” (*terribly disappointed*) would not be annotated manually.

To counter these false negatives, I add a veto stage to the classifier which rejects wrongly classified subjective phrases from the list of objective phrases. This second stage builds a list of sentiment-bearing blocker terms from two additional sentiment dictionaries.

Learner	Precision	Recall	Accuracy
RandomForest	0.984	0.992	0.980
AdaBoost	0.978	0.992	0.975
GradientBoosting	0.978	0.997	0.978
SVC - Linear Kernel	0.965	0.978	0.953
ExtraTrees	0.975	0.980	0.964

Table 3.3.: Performance of different learners for SentiWS + Regression + POS @ 50%. Evalution performed on 20-sentences data set.

GermanPolarityClues [Waltinger, 2010][10] provides lemmata, inflected forms and multi-word expressions. I only use lemmata and word forms as blocker terms. MLSA, the Multi-layered Reference Corpus, [Clematide et al., 2012] provides annotated NP chunks in its layer 2. I extract single sentiment-bearing words from these chunks. Consider the NP "[in der stationären Altenhilfe+]+" (*in stationary care of the elderly*). Here, I extract "Altenhilfe" and disregard the multi-word expression. The token is then used to block subjective expressions as described above with GermanPolarityClues.

Some blocker terms are not useful and are manually whitelisted: "der", "sein", "dem", "er". Another blocker term, "zusammenhangslos" is manually added as it is not found in either sentiment dictionary.

Both blocker terms and candidate terms from the objective list are lemmatized using the `nltk.stem.snowball.GermanStemmer` from the NLTK package [Wagner, 2010]. A blocker term matches only if the full word is found, e.g. "Toll" (*great*) does not block "Tollhaus" (*madhouse*). In a second step, the candidate term is split on the space character. Each token is lemmatized and checked again against the blocker terms. This catches cases like "coolen Sprüchen" because the lemmatizer does not handle the adjective properly in this case. When split, the individual token "coolen" is properly blocked.

Test Questions

I use test questions to test annotator reliability. Test questions are phrases with known answers. As the sentiment annotation task is highly subjective, I set out to find unambiguous test phrases for negative, neutral and positive categories. CrowdFlower recommends 100-200 test questions[11] with a label distribution matching that of the original data.

Negative and positive phrases are obtained from SentiWS. Nouns and adjectives with the highest semantic orientation values, positive and negative, are extracted and reviewed manually. I remove some words with high SO values at my discretion when I consider them ambiguous or otherwise unfitting. For example, "praktisch" (*practical* or *handy*) I consider not strong enough to unambiguous even though the SO value may indicate

[10]Version 2012 downloaded from `http://www.ulliwaltinger.de/sentiment/`

[11]`http://success.crowdflower.com/customer/portal/articles/1365763-test-question-best-practices`, accessed 2014-09-11

otherwise. I create noun phrases from these adjectives and nouns where the combination is semantically useful. Words such as “seine”, “ihre”, “die”, “durch die”, “dank der” (*his*, *her*, *the*, *because of*, *thanks to*) are used to enhance the phrase list. Refer to 3.4 for some examples.

	Noun Phrases
Negative	unerträglicher Unfug
	ein furchtbarer Unfug
	so ein furchtbares Fiasko
	ein furchtbarer Fehltritt
	sein unerträglicher Fehltritt
	eine grauenvolle Pleite
Positive	unschlagbare Genialität
	unvergleichliche Genialität
	seine unschlagbare Genialität
	durch die bewundernswerte Brillanz
	die höchstmögliche Befriedigung
	dank der legendären Genialität

Table 3.4.: Examples of negative and positive noun phrases created from SentiWS.

The neutral noun phrases are generated from a manually created list of neutral determiners, adjectives and nouns. See 3.5 for the specific words. The cross-product between these words results in 294 neutral noun phrases. I add another 22 neutral phrases I consider unambiguous from the 20-sentences data set.

The final set of test questions contains 71 positive, 85 negative and 316 neutral phrases. The neutral phrases dominate the distribution as is the case for both the Socher et al. [2013] and the 20-sentences data set. Distributions for both phrase length and sentiment labels still differ significantly from the original data and run counter to the CrowdFlower recommendation. However, the purpose of these test questions is to be as unambiguous as possible to filter inattentive and otherwise disruptive contributors. This justifies the deviation from CrowdFlower’s recommendation.

The test questions are used in two places. The first place is an initial quiz containing 10 questions for each contributor. If the contributor does not pass the test, then I assume they either do not speak German or do not understand the task. Note that the CrowdFlower task is configured to only allow contributors from Germany, Austria and Switzerland. However, CrowdFlower does not provide the ability to select contributors based on their German skills[12]. Contributors who do not pass the test are barred from working on the task.

After passing the initial quiz, workers will continue to be presented with test questions to check for attentiveness throughout the annotation task. A set of ten annotation tasks is presented on a single page containing one test question.

[12]Crowdflower enabled this feature some time after the annotation process had finished.

Determiners	Adjectives	Nouns
laut der	einzigen	Herbstmesse
der	öffentlichen	Sonderkommission
innerhalb der	kanadischen	Podiumsdiskussion
nach der	gegenwärtigen	Fluggesellschaft
vor der	mexikanischen	Baugesellschaft
während der	spanischen	Landwirtschaft
bei der	diesjährigen	

Table 3.5.: The cross-product of determiners, adjectives and nouns forms the neutral noun phrases.

Financial Considerations

The Institute of Computational of Linguistics at the University of Heidelberg generously provided $200 funding for the annotation. Based on my experience annotating the initial 20-sentences data set, I assume a fast annotator can label 800 phrases in 20 minutes. At 0.004$ per annotation, this results in $9.60 (6.90€) per hour.

To make the best use of the available funding, I employ the subjectivity detection described above.

The final cost is influenced by the amount of data, the frequency of test questions and the number of judgments per unit. I require three judgments per unit, i.e. three judgments from separate contributors for each unit. The CrowdFlower platform computes a final aggregate label from all judgments for a phrase.

CrowdFlower groups tasks into units. By default, five units make up one task and are displayed together on a single page. The contributor is paid a fixed sum per task. One unit per task is always a test question. Accordingly, 20% of cost is attributed to test questions.

I increase the number of units per task to 10 which allocates only 10% to test questions. This enables me to annotate more data for the given funds. As the test questions are merely a test for attentiveness, the reduced frequency is unlikely to impose any problems.

Finally, the CrowdFlower platform imposes a 33% markup on the cost per unit.

Annotation Process

I annotate 592 sentences which consist of 14321 unique phrases. The data is partitioned into three sets: two sets consisting of the first and second 10% of the sentences and a third set containing the remaining 80% of the sentences. The annotation task for a set consists of all unique subtree spans without any previously annotated phrases. Initial results from the smaller sets are inspected for sanity and used to train the subjectivity classifier.

I find some issues in the annotation process after the first set is annotated. The parse trees are not binarized properly. This leads to the inclusion of the tree deepening

Label	HeiST	Stanford Train
Very Negative	0.806%	2.588%
Negative	8.794%	10.786%
Neutral	74.178%	68.989%
Positive	13.605%	13.872%
Very Positive	2.616%	3.764%

Table 3.6.: Fine-grained label distribution.

Label	HeiST	Stanford Train
Negative	9.600%	13.374%
Neutral	74.178%	68.989%
Positive	16.221%	17.637%

Table 3.7.: Coarse-grained label distribution.

mechanism described above [Samuelsson and Volk, 2004].

After the annotation of the first and second data sets, 11500 phrases remain for annotation. I employ two filtering steps to reduce the annotation workload.

The subjectivity classifier categorizes 4241 phrases out of 11989 from the third data set as objective. Note that this includes phrases previously annotated. The following veto stage blocks 1679 out of those 4241 phrases and queues them for manual annotation. A total of 2562 phrases remain considered objective. Subtracting the phrases from the first and second annotation efforts, 9484 segments remain after subjectivity detection.

As a second optimization, phrases which differ only in a leading or trailing comma or period are considered identical. This step removes another 1020 phrases, resulting in 8464 remaining segments. Taken together, both steps remove 3036 phrases (26.4%)

Another issue arises due to a faulty CSV file for the last data set. Improperly quoted phrases are truncated at ",". This leads to duplicate annotations for phrases which are identical after truncation or which only differ in whitespace. In some cases, the duplicate annotations conflict. For these cases, I download the full set of responses from all contributors for that phrase and let the majority vote decide. The sentiment label for all other phrases is assigned by CrowdFlower as an aggregate value of all annotations. A fourth annotation run is necessary to get labels for the full versions of the truncated sentences.

Results and Discussion

The resulting Heidelberg sentiment treebank (HeiST-592) consists of 592 sentences and 14321 unique phrases[13]. The annotation cost \$191.28.

[13]For our NAACL-HLT 2015 submission, we use an extended version of HeiST consisting of 1184 sentences. For convenience, I continue to refer to HeiST-592 as HeiST in my thesis.

Label	HeiST	Stanford Train
Very Negative	3.885%	12.781%
Negative	24.493%	25.960%
Neutral	20.946%	19.007%
Positive	41.892%	27.177%
Very Positive	8.784%	15.075%

Table 3.8.: Fine-grained sentence label distribution.

Label	HeiST	Stanford Train
Negative	28.378%	38.741%
Neutral	20.946%	19.007%
Positive	50.676%	42.252%

Table 3.9.: Coarse-grained sentence label distribution.

Label distributions are shown in tables 3.7 and 3.6 for the fine-grained labels, respectively. The distribution is similar to the Stanford sentiment treebank [Socher et al., 2013]. HeiST has slightly less (3.774 points) negative phrases. Conversely, HeiST has 1.41 points more positive phrases. The difference is mainly shifted to the neutral class which has 5.189 percentage points more in HeiST. Overall, HeiST is slightly more neutral than SST.

Sencentence label distributions are listed in table 3.9 and 3.8. The coarse-grained distributions are very similar between the HeiST and the Stanford sentiment treebank. Neutral sentences make up the smallest portion at around 20%. The German data at 28.378% is less negative than the Stanford sentences (38.741%). Positive sentences provide half of the German data at 50.676% and slightly less of the Stanford data at 42.252%.

CrowdFlower provides several metrics regarding contributor satisfaction and performance. I report the results for the largest, the third annotation job. Results for other jobs are comparable.

Contributor satisfaction is rated on a five-point scale by 18 participants. See table 3.10 for all ratings. The ratings indicate that the task description and the test questions are well-formed and reliable.

Out of 473 test questions, only the positive phrase "unbezahlbare Perfektion" (*invaluable perfection*) is frequently (> 50%) annotated incorrectly. Four out of 53 contributors fail the introductory quiz.

Overall, the high contributor satisfaction and the well-formed sentiment distribution indicate that the annotation task is a success. Training the subjectivity detector on real-world data confirms its usefulness suggested by initial experiments on the 20-sentences data set. Removal of phrases differing only in leading or trailing punctuation further reduces annotation cost. In total, 26.4% cost savings are realized for the third annotation

Category	Rating
Instructions Clear	4.64
Ease of Job	4.4
Test Questions Fair	4.3
Pay	3.8
Overall	4.3

Table 3.10.: Contributor satisfaction on a five-point scale. Survey taken by 18 contributors.

task (21.2% for the full data set). These cost savings are put into perspective by unnecessary or duplicate annotations of segments stemming from incorrect parses, incorrect binarization or bad file formats. In the end, 13133 out of 14321 phrases are annotated in CrowdFlower. Factoring in these engineering mistakes, the resulting savings are a much more modest 8.3%.

These results do not diminish the usefulness of the filtering steps as none of the costly errors stem from filtering. Without filtering, the annotation task would have been more expensive in an academic setting with a small budget. More importantly, savings resulting from filtering scale with data size. Furthermore, the errors support my decision to perform initial annotation runs on smaller data sets to work out problems in the process.

4. Components

In this chapter, I present individual sentiment analysis methods. All methods are evaluated on both the level of individual phrases and full sentences.

I choose to report the macro-averaged F1 measure for phrase-level sentiment analysis. Recall the skewed distribution of the sentiment labels at the node level (see table 3.7). The accuracy measure counts true negatives[1]:

$$\text{Accuracy} := \frac{tp + tn}{tp + fn + fp + tn}$$

A classifier which predicts all nodes as neutral will yield 74% accuracy. Thus, the accuracy measure is too optimistic. Instead, I prefer Precision and Recall which are invariant to change of true negatives [Sokolova and Lapalme, 2009]:

$$\text{Precision} := \frac{tp}{tp + fp}$$
$$\text{Recall} := \frac{tp}{tp + fn}$$

The Fscore measure gives equal weight to precision and recall:

$$\text{F1} := 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

The sentiment analysis task in my thesis consists of classifying a sentence into one of *negative*, *neutral* and *positive*. I calculate the macro-averages over all three classes instead of the micro average [Sokolova and Lapalme, 2009]. The micro-average favors bigger classes while the macro-average gives equal weight to each class. Again, this prevents artificially inflated scores for classifiers which prefer the neutral class. Additionally, the macro-average reflects the desire to obtain good classification performance across all sentiment polarities, even for less frequent classes.

On the sentence level, the class distribution is more uniform. Here, I report the average accuracy over all classes. Macro-average F1 is not a good fit here because some methods fail to return any true positives. In this case, the F1 measure is not defined and average accuracy [Sokolova and Lapalme, 2009] is a better choice:

$$\text{Average Accuracy} := \frac{1}{3} \times \sum_{i \in \left\{\substack{neg,\\ neu,\\ pos}\right\}} \frac{tp_i + tn_i}{tp_i + fn_i + fp_i + tn_i}$$

[1] tp: true positive, tn: true negative, fn: false negative, fp: false positive

In the following sections, the macro-averaged node-level F1 measure is simply called **node F1**. The sentence level average accuracy is referred to as **root accuracy**.

4.1. Sentiment lexicon extraction from weakly-labeled near-domain reviews

Sentiment lexicons are useful tools for sentiment analysis. Several generic lexicons are available. I describe a novel process to obtain a domain-specific sentiment lexicon consisting of a mapping from words to sentiment polarity and strength values.

To obtain the lexicon, I train an unigram regression model on the star ratings of movie reviews. The dimensions in the learned weight vector represent the semantic orientation value of the corresponding word. In this distantly supervised setting, the movie rating serves as a weak label for the SO value of the words contained in a review.

All methods are evaluated by ten-fold cross-validation to avoid bias. The results reported are averaged over all ten test sets. The data is partitioned into 70% train, 10% test, 10% dev and 10% rest[2].

4.1.1. Data Acquisition

Customers write product reviews on Amazon.com. The review text along with a product rating on a five-point scale is published on the product page. I download product reviews for several movies[3] to obtain movie reviews.

I download two set of reviews. The **newreleases** set contains reviews of movies listed as new releases on Amazon.com. This set contains some duplicate reviews as the same movie may be released as both DVD and Blu-Ray. The second set, termed **affordable**, comes from a section on Amazon.de listing DVDs and Blu-Rays costing less than 10€. Corpus statistics are listed in table 4.1.

Data Set	# Movies	# Reviews	# Tokens
newreleases	77	3945	796353
affordable	222	57321	6995448

Table 4.1.: Statistics for movie review corpora.

4.1.2. Domain Adaption

Reviews from Amazon and Film-Rezensionen.de can differ in important aspects. The Film-Rezensionen.de excerpts almost exclusively consist of a short summary of the strengths

[2]The “rest” sets are unused and stem from an earlier mistake in splitting the data. Unfortunately, it is not feasible to re-run all experiments reported in my thesis with 80% train as the training takes several weeks.

[3]Amazon reviews downloader provided by Andrea Esuli, January 2010, `http://www.esuli.it`

and shortcomings of the movie in question. In the Amazon.de reviews, the reviewer also often describes picture and sound quality, features of the collector's edition, shipping speed and other items unrelated to the movie itself.

The following text comes from a Amazon.de review I consider similar enough to Film-Rezensionen.de to be **in-domain**:

> Wenn das wirklich die Fortsetzung von Speed ist, dann erst in der zweiten Stunde des Films. Erst da zeigt deBont, was ihn als Actionspezialist ausmacht. Zwar kommt auch hier nicht die selbe Spannung wie im Vorgänger auf,aber besser als die erste Stunde des Filmes ist sie allemal. Allein schon die Anfangssequenz enttäuscht, und warum wird Sandra Bullocks Rolle mit dieser blöden Fahrschulsequenz in Lächerliche gezogen? [. . .]

The following Amazon.de review snippet is mainly concerned with technical details of a video streaming service and is thus **out-of-domain.**

> Ich bewerte hier nicht die Serie. Das was ich bisher davon sehen konnte war super!Das war leider nicht viel.Ich bin bei Lovefilm (jetzt Prime) und streame die Serie. Furchtbar! Ständig Ladezeiten und Abbrüche. Offensichtlich hat Amazon hier die Technik nicht im Griff. [. . .]

As I am concerned with extracting a domain-specific lexicon for movie reviews, the product features add noise to the lexicon. However, the lexicon extraction requires more data than what can be provided by the original domain.

I adapt the Amazon.de data to the Film-Rezensionen.de by selecting those Amazon.de which closely resemble the FR snippets.

I assume that a review covers one or several topics. For example, the in-domain review has topics such as "Spannung" (*suspense*), "Rolle" (*part*) and "Actionspezialist" (*specialist for action (movies)*). The out-of-domain review covers "Ladezeiten" (*loading time*), "Lovefilm", "streame" (*I'm streaming*). I use the differences in topic distribution between in-domain and out-of-domain reviews to automatically extract in-domain reviews from the set of Amazon.de reviews.

I evaluate the impact of domain adaptation on a sentiment analysis task and in a review score prediction task. The underlying classifier is separately evaluated on a small test set.

Topic extraction and Filtering

The Mallet machine learning toolkit [McCallum, 2002] is used to extract a list of topics from the reviews. The Amazon.de newreleases corpus and the Film-Rezensionen.de snippets are used as input for the training process. Both domains are used to make sure the set of topics can represent (and differentiate) between the domains. Stop words[4] are removed. The tokenizer is configured to extract any continuous sequence of the Unicode letter category as a token. This deviates from the default to include umlauts.

[4]stop-words-german.txt SVN revision 3 from `https://code.google.com/p/stop-words/`

The Mallet training process runs for 2000 iterations. I choose to enable hyperparameter optimization and to learn 300 topics as recommended by Mallet documentation.

After training, reviews are represented in a vector space where each dimension corresponds to a topic. The value of a dimension indicates how prominent that topic is within the document. The document is represented as a distribution over the topic space. This representation is suitable for consumption by standard machine learning algorithms.

I choose a supervised setting to distinguish between in-domain and out-of-domain data. The required training data is obtained as follows:

I select 20 in-domain and 20 out-of-domain reviews from the data. On the in-domain side, 10 items are randomly selected from the Film-Rezensionen.de corpus. The rest is picked manually to closely resemble the FR movie review summaries.

The out-of-domain reviews all come from the Amazon.de newreleases corpus. Two kinds of out-of-domain data are particularly relevant. The first kind contains short, one-line reviews which only comment on one aspect of the product or on the delivery time.

> [. . .] Zur Bewertung möchte ich anmerken, dass Sie unheimlich schnell, auf Bestellungen reagieren, und das alles superschnell hier ankommt. [. . .]

The second kind contains longer reviews which are not primarily about the movie, but rather concerned with the quality of the product like sound production or movie restoration.

> Ich bin ein Blu Ray Freak und liebe es, besonders alte Filme, in hochauflösendem HD zu schauen. Aber bei einem dermaßen zerstückelten Film würde ich sogar lieber ein altes VHS-Band vorziehen!

I manually pick 10 reviews of each class to make up the out-of-domain data set. Additionally, I create a test set out of five in-domain and five out-of-domain reviews from unseen movies solely from the Amazon.de newreleases corpus.

Scikit-Learn is then used to build a SVM classifier with a linear kernel. The classifier yields 100% accuracy on the test set. Results on this small test set are not necessarily very reliable, but the high accuracy is a first indicator that the approach is useful.

The newreleases dataset is divided into 2573 in-domain and 1334 out-of-domain reviews. The set of topics extracted from newreleases is then applied to affordable.

The classifier is trained in the same manner for affordable and then yields 35182 in-domain and 21816 out-of-domain reviews.

To further tune the domain adaptation process, a **n-best** scheme is employed. The classified reviews are ranked according to classifier confidence and **n** most confident predictions are retained. The confidence measure I use is the distance between the vector representation of the review and the hyperplane learned by the SVM classifier. A similar confidence-based data selection scheme is described by Scheible and Schütze [2013].

The influence of domain adaptation and the n-best selection is investigated further in the evaluation section (4.1.5).

4.1.3. Method

A class of learning algorithm exists which uses the l1 norm as regularization. After training, the resulting coefficient vector will be sparse, i.e. few entries will be non-zero. Thus, the l1-regularized learners chooses fewer, more salient tokens which provide the more information about the review score.

In the current setting, I am not interested in using the learner to generate predictions for unseen data. Instead, I view the fitted parameters, i.e. the coefficient vector, as the sentiment lexicon where each dimension represents a token and its positive or negative semantic orientation value. In case of the l1-regularized learners, the sparseness of the vector avoids overpopulation of the lexicon with noisy entries. Thus, the l1-regularized learners provide a form of feature selection. I consider the review score as a weak label for the sentiment orientation value of each word, i.e. the learners are distantly supervised.

I evaluate several classifiers and document representations using 10-fold cross validation. The Mean Squared Error (MSE) loss metric of the predicted review star ratings is used to select the best classifier and representation. I prefer the Mean Squared Error over the Mean Absolute Error as larger errors are penalized more heavily.

As l1-regularized learners, I choose LarsCV, LassoCV, LassoLarsCV [Efron et al., 2004] and ARDRegression provided by the scikit-learn toolkit [Pedregosa et al., 2011] (version 0.14). The "CV" suffix indicates that the learner finds its hyperparameters with an internal cross-validation procedure.

A Support Vector Regression learner with a linear kernel (LinSVR) is included for comparison with the l1-regularized learners.

Reviews are represented in a vector space either as unigrams, bigrams or gappy bigram features. Each dimension either represents a binary presence indicator, the raw occurence count or the TF-IDF normalized count.

The size of the alphabet is controlled by $\mathbf{k}$ which limits how many of the $\mathbf{k}$ most frequent tokens are included.

The search space for the best combination is vast. For this reason, I start with the selection of the best document representation in the vector space. All development and parameter selection is done on the smaller *newreleases* data set.

The parameter $\mathbf{n}$ indicates how many reviews are used in the training process. This parameter is selected in the evaluation section where the larger *affordable* data set is used.

Review representation

The LinSVR and LassoCV learners as two distinct algorithms are used to find the best representation. $\mathbf{n}$ is fixed to 1000 and $\mathbf{k}$ is set to 20000. The results are shown in table 4.2. Binary representation works best in almost all cases. Pang et al. [2002] also report that feature presence instead of feature counts works better.

Learner	Binary	Frequency	TF-IDF
Unigrams - LassoCV	**1.643**	1.870	1.776
Bigrams - LassoCV	1.856	**1.846**	1.880
Gappy Bigrams - LassoCV	**1.950**	1.981	3.806
Unigrams - LinSVR	**1.728**	2.803	2.555
Bigrams - LinSVR	**1.729**	2.019	2.450
Gappy Bigrams - LinSVR	**1.671**	1.904	2.655

Table 4.2.: Influence of different text representations on Mean Squared Error. Bold entries mark the row winner.

Alphabet Size and Feature Sets

Table 4.3 shows predictor performance for various feature sets at $k = 20000$ and $k = 40000$. At $n = 1000$, $k = 40000$ also represents the upper bound of the number of the tokens. The comparison between $k = 20000$ and $k = 40000$ shows how removal of the least frequent 50% of tokens impacts performance. The LinSVR classifier benefits greatly from having all tokens available. LassoCV, on the other hand, learns a sparse weight vector and thus prefers the most salient features. For LassoCV, the only difference occurs in case of the Bigrams + Gappy Bigrams feature set where the smaller alphabet wins.

Features	Learner	$k = 20000$	$k = 40000$
Unigrams		1.728	1.728
Unigrams + Bigrams		1.538	1.466
Unigrams + Bigrams + Gappy Bigrams		1.498	1.485
Unigrams + Gappy Bigrams	LinSVR	1.549	1.485
Bigrams		1.729	1.660
Bigrams + Gappy Bigrams		1.615	1.540
Gappy Bigrams		1.671	1.637
Unigrams		1.643	1.643
Unigrams + Bigrams		1.625	1.625
Unigrams + Bigrams + Gappy Bigrams		1.741	1.741
Unigrams + Gappy Bigrams	LassoCV	1.629	1.741
Bigrams		1.856	1.856
Bigrams + Gappy Bigrams		1.758	1.758
Gappy Bigrams		1.950	1.950

Table 4.3.: Mean Squared Error of different feature sets across alphabet thresholds.

The LinSVR classifier also reacts well to bigger feature sets. Adding bigrams to the unigram feature set improves performance considerably. At $k = 20000$, adding gappy bigrams helps further. LassoCV already performs quite well with the unigram feature set. Adding bigrams or gappy bigrams helps slightly. Unlike LinSVR, LassoCV does not benefit as much from more features due to its preference for more salient features.

Learner Selection

Table 4.4 shows the mean squared error across several learners at $n = 1000$, $k = 40000$. LinSVR yields best performance overall. Among the learners producing sparse coefficient vectors, LassoCV gives the best results.

Learner	Unigrams	Unigrams + Bigrams
LarsCV	1.994	1.994
LassoCV	1.643	1.625
LassoLarsCV	1.910	1.891
LinSVR	1.728	1.466
ArdRegression	2.418	N/A

Table 4.4.: Influence of learner selection on Mean Squared Error.

Number of Features Selected

As mentioned before, I choose l1-regularized learners to select the most salient entries in the sentiment dictionary. A l2-regularized learner like LinSVR creates a lot of noise as shown in table 4.5. LinSVR fits the data best, but produces only very few zero vector dimensions. It is unlikely that 9630 out of 9718 unigrams bear a significant sentiment orientation value. The best performing l1-regularized learner, LassoCV, selects 145 unigrams and likely provides less noisy sentiment dictionaries.

Learner	MSE	Features Extracted
LassoCV	1.643	145
LassoLarsCV	1.910	39
LinSVR	1.728	9630
RandomizedLasso	N/A	147
ArdRegression	2.418	2214

Table 4.5.: Number of features selected by each learner out of 9718 unigrams (n=1000, k=40000) for the last fold in 10-fold CV.

Final Set of Parameters

I choose LassoCV because it produces sparse coefficient vectors at acceptable mean squared error losses. The documents are represented as unigrams with binary vectors.

4.1.4. Evaluation

The evaluation consists of two tasks. One task is a cross-domain rating prediction task to test how well the model generalizes. The second task compares the extracted sentiment lexicon to an existing lexicon.

Any results reported are based on in-domain data. Training data is data selected according to the n-best scheme. For example, $n = 1000$ in a table indicates that the model is trained on the 1000 most confidently classified reviews based on the domain adaptation method.

4.1.5. Rating prediction task on Film-Rezensionen.de data

I train a model on the newreleases Amazon review set and predict the Film-Rezensionen.de star ratings. This is essentially a cross-domain setting.

Table 4.6 shows the results. Performance for LassoCV does not drop considerably when testing on different review data. This indicates the robustness of the select features. LinSVR, on the other hand, performs considerably worse than on in-domain data. This is likely due to over-fitting the data. It is possible that LinSVR learns associations to specific cast members or movie titles. As a first result, I can state that LassoCV is a lot more robust across domains.

Learner	n	k	MSE
LinSVR	1000	20000	1.941
LassoCV	1000	20000	1.622
LinSVR	2573	40000	2.078
LassoCV	2573	40000	1.483

Table 4.6.: MSE when testing on the Film-Rezensionen.de data set.

Figure 4.1 shows the effects of the domain adaptation. The LassoCV learner is trained on the affordable data set and tested on Film-Rezensionen.de. The figure shows that the reviews considered unfit for the Film-Rezensionen.de task actually produce a better classifier.

4.1.6. Sentiment Analysis task

I compare the performance of the learned domain-specific sentiment lexicon against SentiWS, a publicly available domain-independent lexicon. I report the F1 measures [Sokolova and Lapalme, 2009] for both positive (F_{pos}) and negative (F_{neg}) classes.

Evaluation is done with the 20-sentences test set. The goal is to accurately predict the correct label (negative, neutral, positive) for each subtree span in the tree.

The subtree span is tokenized. The tokens are then looked up in the sentiment lexicon. The final sentiment orientation value for a phrase is the sum of the sentiment orientation values over its tokens.

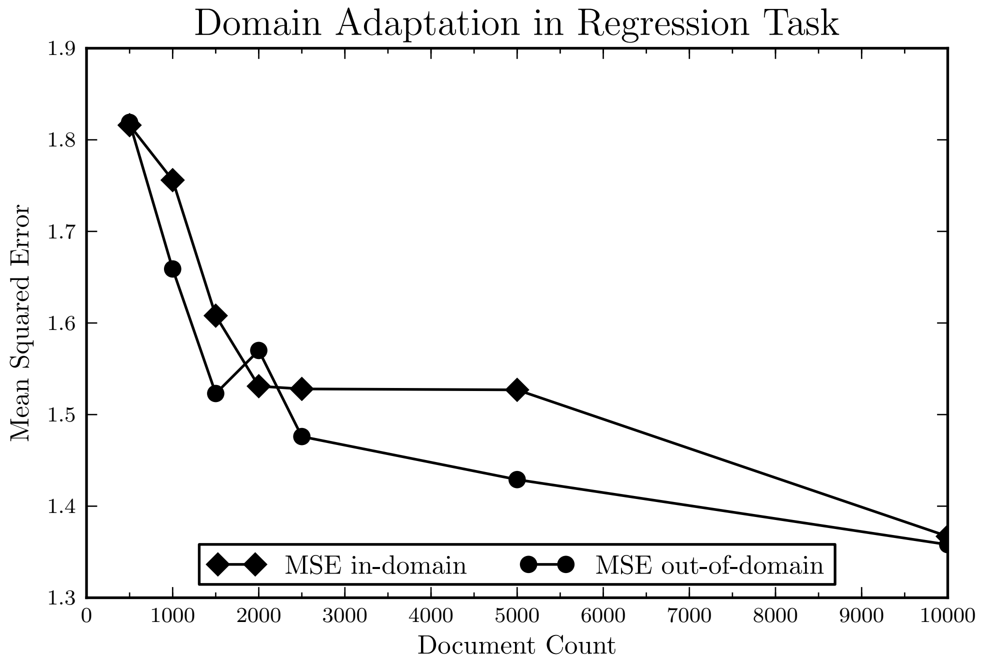


Figure 4.1.: Comparison between random in-domain and out-of-domain documents (MSE).

The sentiment lexicons are extracted from the affordable dataset. Alphabet size k is set to 80000 to include as many tokens as possible. Hapax legomena are removed from the alphabet to speed up learning

Figure 4.2 compares models trained on in-domain and out-of-domain reviews. The reviews are sampled randomly (without replacement) and thus not selected according to the n-best scheme. Figure 4.2 does not show a superiority of the in-domain data. Similar to the results shown in the regression task (figure 4.1), the out-of-domain reviews provide better performance in the majority of cases.

The introduction of the n-best scheme improves the situation. Figure 4.3 shows a comparison between n-best and random selection schemes on the affordable data set. It is clear that selecting the most confidently predicted in-domain reviews as opposed to random reviews improves performance considerably. In particular, the n-best scheme outperforms both randomized in-domain and out-of-domain up to $n = 2500$ and ties for higher n.

Figure 4.3 also indicates that $n = 1500$ performs best.

For comparison against SentiWS, the numbers reported are averages over the tests sets from the ten-fold cross-validation version of HeiST. Four SentiWS variants are presented. In addition to the naive sum over word weights, the *flip* variant implements a simple negation handling scheme similar to the *vote+flip* method described by Choi and Cardie [2008]. The list of German negator terms is taken from Clematide and Klenner

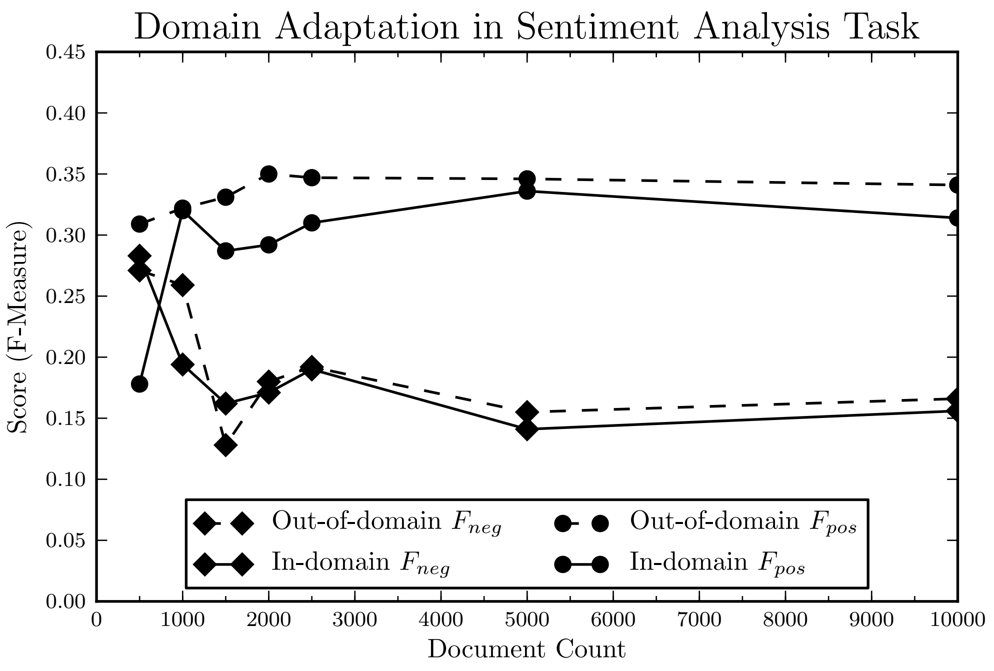


Figure 4.2.: Comparison between in-domain and out-of-domain documents (F1)

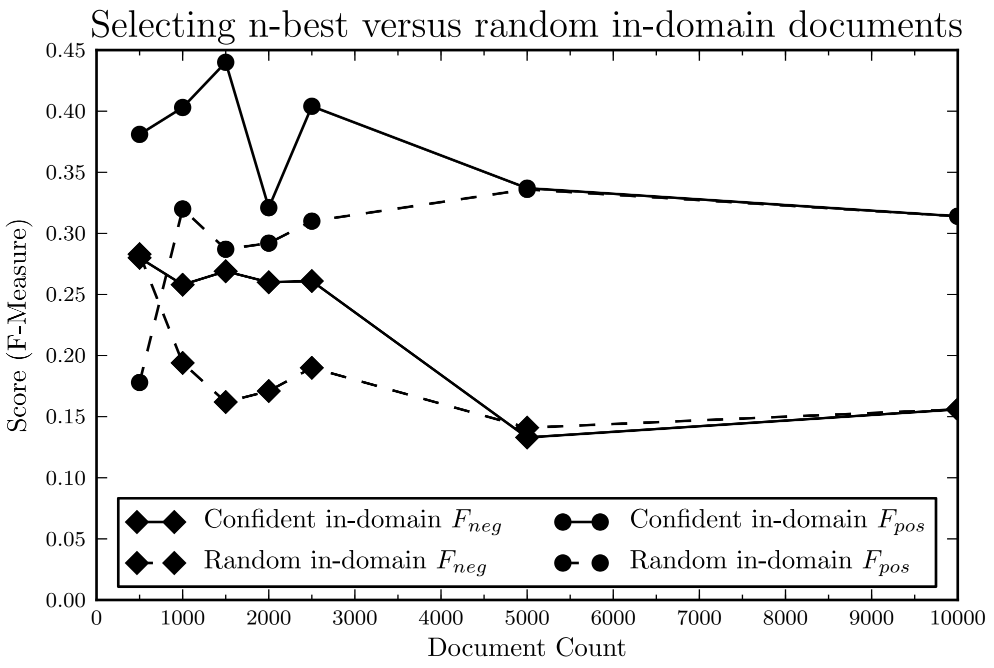


Figure 4.3.: Comparison between most confident in-domain and random in-domain reviews.

[2010]. The *POS* variant matches the POS tags of the SentiWS lemmata against the tags found in the HeiST parse trees.

The learned dictionary is extracted from the most confident $n = 1500$ from the newreleases corpus. Results are shown in table 4.7. In comparison to the LinSVR learner, the sparse lexicon produced by LassoCV performs best for the all-phrase task. The LinSVR-based lexicon slightly outperforms LassoCV on the full sentence task. A possible explanation is the bigger coverage which allows better performance on longer items such as full sentences. However, SentiWS clearly outperforms the learned lexicons by a large margin.

System	Node F1	Root Acc
SentiWS	0.624	0.711
SentiWS Flip	0.622	0.689
SentiWS POS	0.600	0.688
SentiWS Flip+Pos	0.602	0.675
Lasso	0.391	0.569
Lasso Flip	0.382	0.563
LinSVR	0.217	0.618
LinSVR Flip	0.225	0.631

Table 4.7.: Comparison between domain-specific learned sentiment lexicon and SentiWS.

4.1.7. Discussion

I propose a novel, distantly supervised approach to domain-specific sentiment lexicon extraction on movie review data using a l1-regularized learner. I evaluate the proposed approach on a score prediction task and on a sentiment analysis task. The results of the score prediction task show that the l1-regularized learner does not overfit the original data set and performs better on other movie review data. The evaluation further shows that the automatically extracted lexicon is useful in the sentiment analysis task. In particular, using only the most confident predictions provided by the domain adaptation step improves performance considerably. I compare the proposed approach to SentiWS, an off-the-self domain-independent sentiment lexicon. SentiWS yields superior results compared to the automatically extracted lexicon. In chapter 5, I describe and evaluate a method to combine the domain-independent SentiWS lexicon with the new domain-dependent lexicon.

The results also show that the number of documents influences the number of extracted words in the lexicon. A possible explanation is the α parameter used by LassoCV which influences the density of the weight vector. The cross-validation procedure to set α yields less dense weight vectors if more samples are given. The cross-validation procedure aims to find the α yielding the smallest MSE rate. As shown in figures 4.1 and 4.3, MSE rate, albeit useful in guiding early modeling decisions, is not a good proxy for performance on the sentiment analysis task. In particular, sentiment analysis suffers as MSE goes down with increasing n and vice versa. This observation leads me to the hypothesis that optimizing α directly on the F-scores in the sentiment analysis task instead of MSE should yield better results. In particular, a larger set of tokens learned from a bigger set of reviews would improve coverage.

4.2. Projection

Cross-Lingual Sentiment Analysis leverages resources from a source language to enable sentiment analysis in a resource-deprived target language. A recurring theme in the literature is the use of machine translation methods [Banea et al., 2008, Denecke, 2008a,b]. By translating the data from the target language to the source language, tools and resources from the source language are made available for sentiment analysis. The sentiment labels are then projected back to the source language. For data units consisting of single sentences or documents, the projection is straightforward as there is a one-to-one mapping between units in source and target language. In case of the RNTN [Socher et al., 2013],a sentence is divided into smaller subsentential units. Every node in a parse tree yields an unit and there is no inherent, one-to-one mapping between parse nodes in source and target languages. This complicates the task and potentially compounds the accuracy loss imposed by the translation engine. Tiedemann [2010] provides a supervised toolkit for automatic alignment of nodes between parse nodes. I use automatically generated node alignments to project the sentiment labels from the source language to the target language.

Thus, my novel contribution is the extension of existing cross-lingual sentiment analysis methods to the tree-based RNTN.

I evaluate the novel sentiment label projection method on HeiST. In a future section 4.3, I compare the novel method to a RNTN trained directly on the german sentiment treebank. The goal of the evaluation is to show the general viability of cross-lingual sentiment analysis based on the RNTN. Additionally, I aim to evaluate the arguments against cross-lingual sentiment analysis by Balamurali et al. [2013] in the RNTN setting where annotation is more expensive. The hypothesis is that the novel method works, but may suffer some additional performance losses due to the tree-to-tree alignment required for label projection. As for the cost factor, I expect the method to be more competitive as re-using existing corpora in the source language presents larger savings than for simpler, sentence-based methods.

4.2.1. Method

Translation

I use the Google Translate service to translate the full set of reviews downloaded from Film-Rezensionen.de (see section 3.1). Each review is split into individual sentences using the German Punkt sentence tokenizer shipped with NLTK 2.0.4 [Wagner, 2010]. Each resulting sentence is translated using the Google Translate API from German to English. Funding for the translation API was provided by my supervisor, Dr. Yannick Versley.

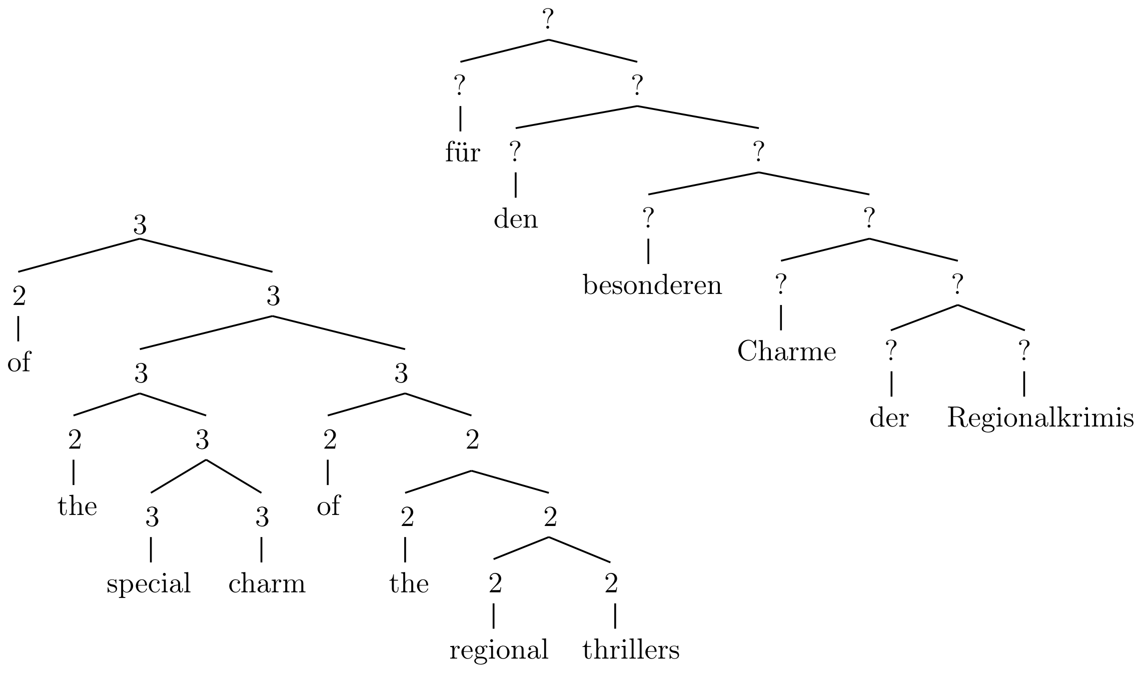


Figure 4.4.: Unaligned trees with sentiment labels on the English side.

The HTML tags `<em>` and `<b>` are occasionally used to indicate movie titles. I opt to replace these with double quotes instead of removing the tags completely. For movie titles like "Stirb Langsam", Google Translate recognizes the named entity and does not wrongly incorporate the individual tokens into the sentence structure. In fact, Google Translate provides the proper English title for the movie (*"Die Hard"*). Later on, the quotation marks are removed for projection. The NLTK sentence splitter produces different results for quoted and unquoted text. I apply the segmentation produced for the quoted text to the unquoted version as well for consistency.

I use the RNTN implementation [Socher et al., 2013] shipped with CoreNLP release 2014-01-04 [Manning et al., 2014]. The English RNTN model is also provided by the CoreNLP toolkit and trained on the Stanford sentiment treebank.

Although I translate the full set of movie reviews, the data of interest is the subset constituting the HeiST. The translated counterpart of these sentences is processed with the RNTN. On the German side, the sentences are already parsed as described in section 3.1.1.

Figure 4.4 shows the starting point. The sentiment labels for the translated phrase are have been obtained from the RNTN. The labels for the original phrase and its segments are still unknown and need to be projected. Thus, the next step is the alignment of the parse tree nodes. Note that the parse trees are already binarized and unary nodes are collapsed as this is the format required by the binary composition function implemented by the RNTN.

Alignment Model

Projecting the sentiment labels from English to German requires an alignment between the parse trees where corresponding nodes are mapped. Manual creation of the alignment is possible, but the goal is to avoid manual annotation efforts. Tiedemann and Kotzé [2009] present a supervised method to align parallel treebanks. The resulting toolkit is called Lingua-Align [Tiedemann, 2010] and provides a rich set of feature extractors. I use version 0.04 of Lingua-Align. I modify some minor aspects of the tree reading code for compatibility with my parse trees. Jörg Tiedemann kindly incorporated my changes into Lingua-Align 0.05 [5].

The SMULTRON corpus [Samuelsson and Volk, 2006, Volk et al., 2010] consists of four treebanks generated from a novel, economy texts, a technical manual and alpine club periodicals. The corpus provides treebanks in English, German, Spanish, Swedish and French. Not all text domains and alignments are available in every language pair. For the German-English language pair, three domains (novel, economy, manual) consisting of 500 manually aligned sentences are available. I use the concatenation of these treebanks and corresponding alignments as the training data. Training on all data ought to provide a robust, accurate classifier.

Feature Sets I opt to construct two feature sets for Lingua-Align. The **simple** feature set consists of features described in the official Lingua-Align tutorial. The **complex** feature set contains more features and comes from the "europarl" example shipped with Lingua-Align. The inclusion of two different feature sets is mainly due to reduced training time for the smaller set.

Table 4.8 lists the features for both sets. The subtree features are derived from shape or location of a given subtree. The *treelevelsim* feature, for example, measures the similarity between the relative height of two different sub trees as determined by the distance to the parent node divided by total tree height. Label features come from syntactic categories or part-of-speech tags. The *catpos* feature extracts a binary feature for the category for target and source links. If the source node is labeled NN and the target node is labeled NP, the resulting feature will be *catpos_NN_NP => 1.* Word alignments come from external sources and can provide either binary features to indicate links between terminal nodes or a percentage describing subtree overlap.

Additionally, features can be modified to take into account the tree structure. By prefixing a feature name with *parent_*, the value for the parent node is extracted. The prefixes *children_* and *sisters_* are also allowed in which either the combination of all features is generated or the largest value is used.

Finally, features can be multiplied (indicated by `*`), averaged (+) or concatenated (`.`). As an example, the feature *treelevelsim.catpos* extracts the node label for both source and target candidate node. The resulting binary feature is then multiplied with the tree level similarity between the candidate nodes and the resulting feature might be *treelevelsim_-catpos_NN_NP => 0.123.* The Lingua-Align documentation, in particular the man page for `Lingua::Align::Features`, describes the features in more detail.

[5] `https://bitbucket.org/tiedemann/lingua-align/downloads`

Feature Category	Simple	Complex
Subtree Feature		nrleafsratio
	treelevelsim	treelevelsim
	treespansim	treespansim
		treespansim*treelevelsim
Label	catpos	catpos.parent_catpos
	parent_catpos	parent_catpos
	children_catpos	
Word Alignment	moses	moses
	children_moses	children_moses
	parent_moses	moses.catpos
		moseslink
Combined		treespansim.catpos
		treelevelsim.catpos
		moses.parent_moses
		sister_moses.catpos

Table 4.8.: Feature Sets for Lingua-Align

Word Alignments Both feature sets integrate existing word alignments as the Lingua-Align documentation indicates that these features help performance. I use PostCAT [Graça et al., 2009] version 2 to generate the word alignments. PostCAT provides several word alignment models and alignment decoding options. I choose the agreement model and the posterior decoding algorithm as it has been shown to perform well on the en-de language pair [Ganchev et al., 2008].

I train the model on the en-de portion of the EuroParl corpus (version 7) with maximum sentence length 40. Alignments are symmetrized with the intersection heuristic. Prior to predicting the alignments, the data is tokenized with the *tokenize-anything.sh* script distributed with CDEC [6] (version 2013-07-13). and lemmatized with the lemmatizers provided by the mate-tools[7].

Alignments are then created for both the SMULTRON corpus and the Film-Rezensionen.de data.

Predicting node alignments

After the alignment model is trained, the node alignments between the original German and the translated English movie reviews need to be predicted. The parse trees are binarized as required by the RNTN.

The German parse trees are also deepened as described earlier [Samuelsson and Volk,

[6] `http://www.cdec-decoder.org/`

[7] `https://code.google.com/p/mate-tools/`

2004] as the SMULTRON corpus follows the same conventions. These steps ensure that the data is as close to the training data as possible to obtain accurate predictions. Additionally, this improves the binarization process as described above.

Similar to the training step, the prediction step also makes use of externally supplied word alignments.

Lingua-Align only supports 1-to-1 alignments. This leads to missing alignments in some cases, e.g. for idiomatic expressions or compound nouns where $1 : n$ or $n : m$ mappings would be required. The binarization step is another source for missing alignments as it inserts additional nodes with artificial labels. These labels are not seen during the training process. Additionally, introducing new nodes influences subtree-based features irregardless of the node label. Thus, the new nodes are likely harder to correctly align and may end up unaligned.

In the next section, I detail two methods to handle missing alignments during the sentiment label projection.

Projecting sentiment labels

Projecting the sentiment labels is straightforward once the node alignment is available. However, not all nodes are aligned. Two strategies are available for unmapped nodes. The **default** strategy simply assigns *neutral* as the default label to unaligned nodes. The **ancestor** strategy infers the sentiment label from a node's parent. If no suitable parent is found, then the default *neutral* is assigned.

An additional strategy is available for root nodes as they have no ancestors. The **implicit** strategy assumes an implicit link between root nodes; that is, the sentiment labels for the full sentences should be equivalent. The implicit link is only assumed if no explicit link for the target root node is provided by Lingua-Align. Note that **implicit** may be combined with either **default** or **ancestor**. Labels projected via **implicit** may serve as parents for **ancestor**.

Lingua-Align provides two confidence classes for alignments. *Good* alignments are based on confident predictions made by the underlying classifier. *Fuzzy* alignments are less certain.

Figure 4.5 shows a final projection. Alignments for non-terminal nodes are marked with bold arrows. Please note that alignments between terminal nodes are also predicted by Lingua-Align even with external word alignments already available. Nodes suffixed with *ANC* indicate nodes for which the **ancestor** strategy is applied. In the current example, this inheritance heuristic works well.

4.2.2. Evaluation

I do not evaluate the alignments produced by Lingua-Align separately. Instead, I focus on the sentiment analysis task where the correct label for each node must be predicted. This task is dependent on good node alignments, but good node alignments do not guarantee the validity of the general method.

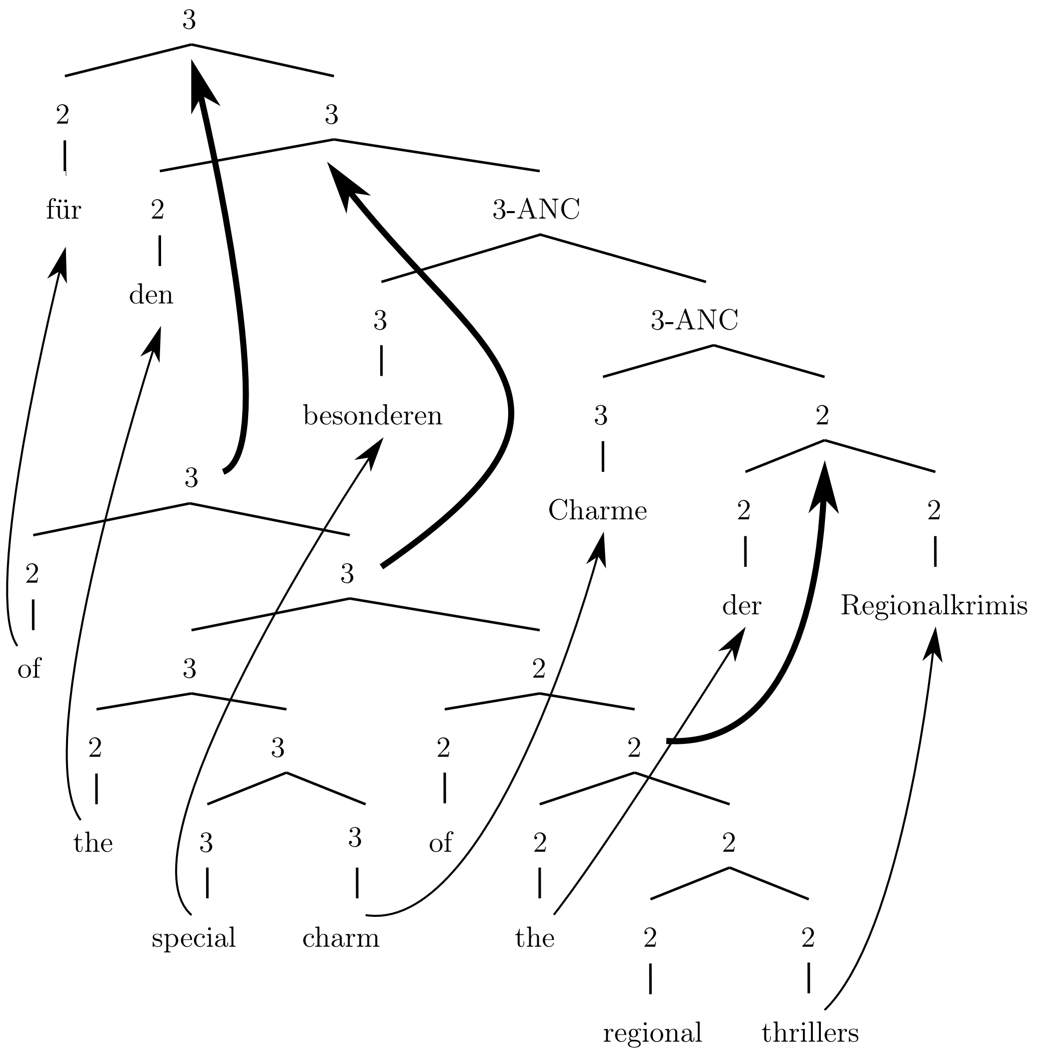


Figure 4.5.: Projecting the sentiment labels.

In this evaluation, I also compare the different label projection strategies for unaligned nodes. The evaluation is based on the same splits used throughout my thesis for the ten-fold cross-validation runs. For each fold, only the test set is used as the projection step itself does not require training data. Although fine-grained labels based on the $[0; 5]$ scale are projected, the evaluation considers only coarse-grained labels. Thus, *slightly negative* and *very negative* are both mapped to *negative*. The same mapping is applied to the positive labels.

Table 4.9 shows the results.

Alignment Model	Projection Mode	Link confidence	Node F1	Root Acc
Complex	Inherit	Good	0.572	0.742
		Good & Fuzzy	0.572	0.742
	Default	Good	0.572	0.742
		Good & Fuzzy	0.572	0.742
	Implicit	Good	0.572	0.742
		Good & Fuzzy	0.572	0.742
Simple	Inherit	Good	0.591	0.750
		Good & Fuzzy	0.591	0.750
	Default	Good	0.591	0.750
		Good & Fuzzy	0.591	0.750
	Implicit	Good	0.591	0.751
		Good & Fuzzy	0.591	0.751
Amazon Lasso			0.391	0.569
SentiWS			0.624	0.711

Table 4.9.: Evaluation results for Projection.

4.2.3. Discussion

Across all available parameters for the Projection method, the results are very close. The biggest influence on performance is the choice of the underlying alignment model for Lingua-Align. The **simple** model performs better than **complex**. Both models leave 330 out of 22822 nodes unaligned on the target (German) side. For these nodes, the **default** and the **ancestor** strategies heuristically apply labels. Both strategies perform equally well. Adding **fuzzy** links, i.e. alignments with less confident, does not change performance in any way.

The **implicit** projection mode offers a minor improvement for the simple model. Apparently, the simple model leaves the target root node unaligned in at least one case. Instead of assigning the default of *neutral*, the **implicit** mode correctly projects the root label. Accordingly, only 329 instead of 330 nodes must be handled by **default** or

ancestor strategies.

At 75.1% accuracy on the sentence level, the projection method performs strongly. The majority-class baseline ($\sim$ 50% accuracy, all positive) is easily beaten. The projection method easily outperforms the regression-based sentiment dictionary (subsection 4.1.7). SentiWS is also beaten by a small margin. Across all nodes, F1 0.592 also indicates strong performance. SentiWS slightly outperforms projection in the all-node setting.

4.2.4. Summary

I present a novel method for cross-lingual sentiment analysis in a subsentential setting. In addition to losses suffered by traditional machine-translation based CLSA methods, the automatic parse tree alignment adds further noise to the process. From the results, it is clear that the method works and performs quite strongly. Comparison with other methods in coming chapters will investigate the competitiveness, financially and performance-wise, of the projection approach.

Out of several parametrizations of the projection process, the quality of the underlying alignment model seems to be of great importance. Adding an implicit link between target and source root nodes is likely to increase performance without negative effects.

The strong influence of the alignment model opens a further possibility of tuning the model. Instead of using expensive sentiment treebank to develop and test the system, it ought to be possible to automatically select the best feature set for the alignment model using held-out data (aligned treebank data) when training the model on an aligned treebank. Improved alignment quality should result in improved projection results. This hypothesis needs further validation, however. Quite often, upstream improvements in NLP do not yield equivalent downstream improvements if at all (see MSE vs lexicon quality in 4.1.7 or the lack of influence of improved word alignments on BLEU score [Ganchev et al., 2008]).

In consideration of Balamurali et al

Balamurali et al. [2013] argue against the usefulness of cross-lingual sentiment analysis methods. For supervised methods in particular, the authors state that the creation of annotated corpora in the target language is cheap. They argue that no more than 500 documents are required for good performance. Such small corpora are easy enough to annotate without having to resort to noisy cross-lingual methods.

Their arguments are certainly reasonable considering regular corpus-based sentiment analysis where a supervised machine learner is trained with document-level annotation.

In case of the RNTN, the creation of the training corpora is much more expensive as a single sentence yields several data units. From the results by Socher et al. [2013], it is known that the RNTN beats methods like the naive system described by Balamurali et al. [2013].

Generally speaking, Balamurali et al. [2013] assume that their saturated system already provides acceptable performance at 500 sentences. What happens if more accuracy is desired? As the system is saturated, a different sentiment analysis needs to be used which might require a lot more training data (such as the RNTN). In particular, the German system by Balamurali et al. [2013] yields about 75% accuracy which might not be enough. It has been shown that using bigger feature sets (bigrams, trigrams, POS tags) in addition to unigram features improves performance for sentiment analysis [Atalla et al., 2011, Wang and Manning, 2012]. As bigrams are more sparse than unigrams, more training data is needed.

Another issue which limits applicability of the findings by Balamurali et al. [2013] is the granularity of the annotation. As a middle ground between the subtree-level annotation of the RNTN and document-level annotation used by the authors, some methods perform sentiment analysis on the sentence level. As the number of tokens per sentence is typically smaller than for a full movie review, more units need to be annotated. On the other hand, time per unit goes down.

Balamurali et al. [2013] also show that low-quality translation systems (i.e. simple dictionaries) are not adequate for CLSA. From their results, they deduce that MT systems do not adequately capture the cultural differences in sentiment expression and that MT systems introduce noise into the data through inaccurate translations.

From their data, they offer the example *This X sucks.* Their MT system, Microsoft Translate, provides a literal translation to the french verb *sucer*, which is nowhere to be found in the French data.

I repeat the *This X sucks* experiment by translating *This movie sucks.* to French. The resulting phrase with Google is *Ce film est nul.* which is a correct and fluent translation. Early in the research for the projection system, I used some sample sentences from the data to evaluate both Google Translate and Microsoft Translate. I found the translations provided by Google to be of superior quality and I gladly asked my supervisor to provide the funding for the Google Translation service instead of using the free Microsoft offering.

From these observations, I would like to propose that the findings reported by Balamurali et al. [2013] do not necessarily reflect state-of-the-art in machine translation. Naturally, both companies continue to improve their offerings and movie review translations are not necessarily indicative of overall performance.

In summary, the different, more fine-grained nature of the data used by the RNTN and the differences in the machine translation system suggest that the restrictions reported by Balamurali et al. [2013] do not apply in this setting.

4.3. Heidelberg Sentiment Treebank & RNTN

The RNTN [Socher et al., 2013] is originally trained on the Stanford sentiment treebank which contains 11945 sentences. The Heidelberg sentiment treebank is smaller at 592 sentences, mainly because the annotation process is expensive. In this subsection, I investigate the performance of the RNTN when trained on the German sentiment treebank. Additionally, I present and evaluate two methods to increase performance for the given data set.

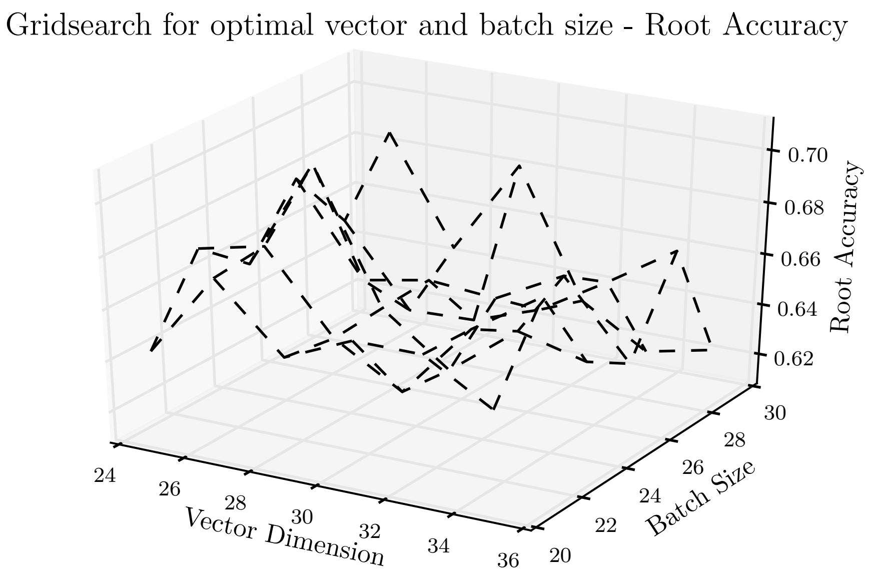


Figure 4.6.: Testing root accuracy at various batch sizes and vector dimensions.

4.3.1. Hyperparameter selection

The RNTN has two hyperparameters which can influence classifier performance. The first one is the vector size. The other hyperparameter is the number of sentences per training batch (**batchSize**). Socher et al. [2013] indicate ranges between 25 and 35 dimensions for the vectors and between 20 and 30 for the batch size. Within these boundaries, I perform a grid search to find the best set of parameters. The hyperparameters are tested on the development sets.

In general, the model is trained with the `edu.stanford.nlp.sentiment.SentimentTraining` class shipped with CoreNLP [Manning et al., 2014] version 2014-01-04. The random seed for the training process is set to 250891. Training runs for 400 epochs. All evaluation measures are calculated as described at the beginning of chapter 4.

Individual results from each split in the ten-fold cross-validations are averaged to produce the final result. As the grid search tests many parameter combinations, I use the GNU `parallel` tool [Tange, 2011] to spread the computation across multiple compute nodes.

Figures 4.6 and 4.7 shows a plot of root label accuracy and node F1 at vector dimensions $[25, 27, 29, 31, 33, 35]$ and batch sizes $[20, 22, 24, 26, 28, 30]$. Several hyperparameter combinations yield good results. I choose 22 as batch size and 25 as vector dimension.

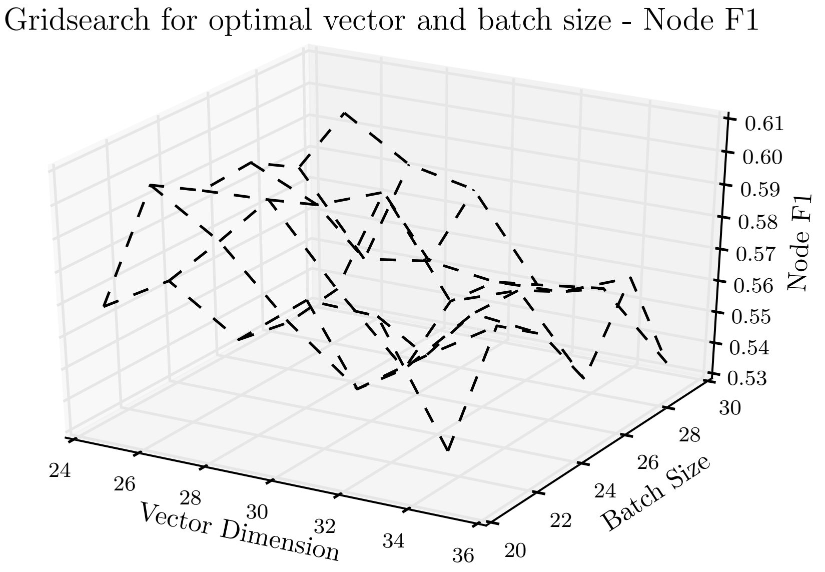


Figure 4.7.: Testing node F1 at various batch sizes and vector dimensions.

4.3.2. Impact of training data on performance

Figure 4.8 shows the impact of the size of the training set on classifier performance. The RNTN is trained on the first $[500, 1000, \ldots, 8500]$ sentences of the Stanford training set and evaluated on the full test set.

The curve gives an approximation of the performance on the German sentiment treebank, although the German data is different from the English data. Increasing the size of the training data from 500 to 85000 sentences improves performance by 25.04% for root accuracy and 36.4% for the F1 measure.

I perform a linear regression analysis on the data reported in figure 4.8 using the `polyfit()` in numpy 1.8.1. For all-node F1, the resulting equation

$$F1(n) = 2.349^{-5} \times n + 0.583$$

indicates a performance gain of 0.0118 points for every additional 500 sentences. For root label accuracy,

$$Acc(n) = 2.0023^{-5} \times n + 0.5914$$

shows an improvement of 0.0100 points per added 500 sentences.

These figures are rough estimates. Around 6000 sentences, performance gains start to slow down. Additionally, the estimates are only valid for the ranges from which the equations are estimated. 100% accuracy is very unlikely and the curve will asymptotically approach some upper bound below 100%. However, the estimates are helpful in

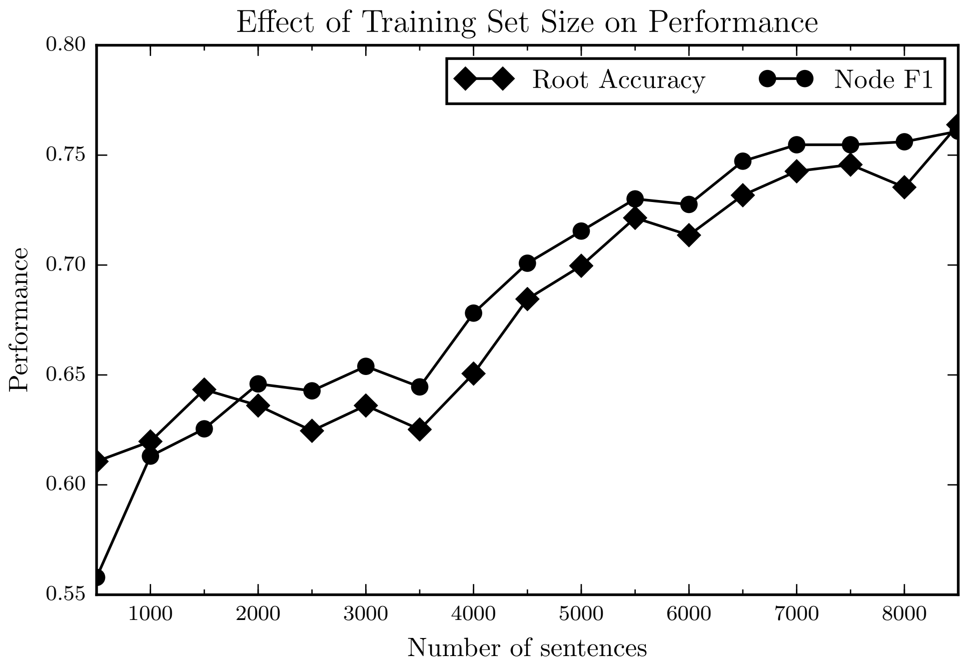


Figure 4.8.: Performance at various training set sizes.

estimating the effect of any performance improvements on cost.

Evaluation of plain RNTN

Table 4.10 shows the performance of the RNTN when trained on the Heidelberg sentiment treebank. Previous systems are listed for comparison purposes. The last rows show values obtained by training and testing the RNTN on the English Stanford Sentiment Treebank at 500 and 1000 sentences. As a very basic check, the values for German and English are close enough to conclude that there is no gross error in setting up the Heidelberg Sentiment Treebank or the RNTN.

System	Vector Size	Batch Size	Node F1	Root Accuracy
RNTN Tuned + HeiST	25	22	0.590	0.679
Projection	–	–	0.591	0.751
Amazon Lasso	–	–	0.391	0.569
SentiWS	–	–	0.624	0.711
RNTN + SST @500	25	27	0.558	0.611
RNTN + SST @1000	25	27	0.613	0.62

Table 4.10.: Performance for RNTN and other systems.

Discussion

The RNTN is originally developed by Socher et al. [2013] on English data. Training the RNTN on German data is straightforward as it simply requires annotated data. The performance when trained on a small data set is lacking, however. Both the projection method and SentiWS are cost less and produce better results. In case of SentiWS, its lack of complexity is an additional point in favour of the lexicon. In the next section, I will describe some ways to enhance performance without providing more data.

4.3.3. Enhancing the RNTN with Word Clusters

More data often yields better results. As shown above, the RNTN benefits from having more data available. A huge contributor is the problem of unseen data. Unknown words in the test data default to the *neutral* class, which is a problem for the RNTN as the bottom-up prediction process depends on correct leaf labels.

A simple way of alleviating the problem of data sparsity is described by Popat et al. [2013]. By replacing individual words with word clusters, they improve the performance of their system in a data-sparse setting. As word clusters can be cheaply obtained from unlabeled data, this improvement is very viable.

Method

The cluster-based enhancement is implemented as a preprocessing step for the RNTN.

I extract Brown clusters [Brown et al., 1992] using the `brown_cluster` utility provided by Liang [2005][8] from two different corpora. One corpus is the generic newspaper based TüBa-D/Z corpus [Telljohann et al., 2004] from which $c = 1000$ clusters are learned. However, there is likely a large vocabulary gap between newspaper and movie review domains. This leads, in theory, to a large amount of unseen words when applying the clusters to the review. This is the exact problem I hope to solve by using the clusters.

For this reason, I extract clusters from another movie review corpus which likely is closer in vocabulary to the Film-Rezensionen.de reviews used for the Heidelberg sentiment treebank. The reviews come from the *affordable* Amazon data set described in section 4.1. The full subset classified as in-domain is used, consisting of 35182 documents.

Individual clusters can be merged together based on a cut-off value. Each cluster is identified by a binary string. The cut-off value specifies how many of the most significant bits are retained. Any clusters which differ in bit positions pruned by the cut-off value are merged. As the merging is based on less significant bits, closely related clusters are merged first.

Some words are so infrequent that they cannot be reliably assigned to a cluster. A frequency threshold is used to filter out infrequent and thus unreliable words.

The preprocessing step then consists of replacing each word in the Heidelberg sentiment treebank with the ID of its most likely cluster.

[8]`https://github.com/percyliang/brown-cluster`

In preliminary experiments, I test the influence of different word classes. I run three experiments where the clusters are only applied to Verbs, Nouns or Adjectives+Adverbs. The results are not promising, with performance below full cluster application. I also test the different preprocessing options in other preliminary experiments. Stemming and/or lowercasing to improve matching actually decreases performance. I do not pursue any of these modifications any further.

Evaluation

The goal of the evaluation is to find set of parameters which yield the best performance. For this reason, I test clusters based on TüBa-D/Z and Amazon data and various cut-off values and frequency thresholds.

Table 4.13 shows the results for the Amazon-based clusters at c=5000 and various values for frequency threshold and number of significant bits.

Table 4.12 and 4.11 list more results. For Amazon, 10 is introduced as an additional value for frequency threshold as the overall word frequencies in the smaller Amazon *affordable* corpus are lower.

Freq Thresh	Significant Bits	Clusters	Node F1	Root Acc
0	8	181	0.489	0.667
	9	294	0.514	0.640
	10	434	0.511	0.649
	11	591	0.525	0.659
	12	725	0.513	0.627
	13	822	0.528	0.662
	14	903	0.528	0.644
	15	948	0.538	0.648
30	8	181	0.506	**0.672**
	9	294	0.509	0.649
	10	434	0.520	0.667
	11	591	0.518	0.671
	12	725	0.533	0.649
	13	822	**0.544**	0.665
	14	903	**0.550**	**0.690**
	15	948	**0.547**	**0.670**

Table 4.11.: Results for TüBa-based clusters (c=1000).
Bold entries mark three winners per column.

Discussion

The number of significant bits and thus the amount of clusters is a big influence on performance. As a general trends, having more clusters improves performance. A lower

Freq Thresh	Significant Bits	Clusters	Node F1	Root Accuracy
0	8	217	0.500	0.670
	9	351	0.520	0.660
	10	509	0.541	0.664
	11	629	0.547	0.654
	12	735	0.538	0.637
	13	837	0.571	0.661
	14	927	0.561	0.661
	15	985	**0.578**	0.664
10	8	217	0.517	0.657
	9	351	0.525	0.663
	10	509	0.535	0.644
	11	629	0.548	0.658
	12	735	0.560	0.652
	13	837	0.566	0.676
	14	927	**0.576**	0.660
	15	985	**0.575**	0.644
30	8	217	0.525	0.663
	9	351	0.531	0.663
	10	509	0.537	0.667
	11	629	0.542	0.658
	12	735	0.566	**0.686**
	13	837	**0.575**	**0.677**
	14	927	0.562	0.650
	15	985	0.570	**0.686**

Table 4.12.: Results for Amazon-based clusters (c=1000).
Bold entries mark three winners per column.

number of significant bits resulted in more merged clusters. The resulting large clusters likely are too generic to adequately capture sentiment information.

In particular, more than c=1000 clusters are required for good performance. c=5000 provides notable performance improvements and also requires a higher number of significant bits to represent the cluster IDs.

The cluster domain is also important. Comparing clusters obtained from a generic newspaper corpus (table 4.11) and a movie review corpus (table 4.12), the in-domain clusters yield slightly better performance.

The frequency threshold also positively influences performance.

The TüBa-based clusters benefit from having infrequent words removed (table 4.11). On the *affordable* corpus derived from Amazon, the benefits of the frequency threshold are not as pronounced. Additionally, both Amazon-based runs suffer slightly when the frequency threshold is too high at 30. Both perform better at a threshold of 10.

A possible explanation for the bigger improvement in the TüBa-based clusters is the size of the underlying corpus. Larger corpora such as TüBa-D/Z will have more infrequent words (Zipf's law) which can contribute noise to the clustering process. In addition, differently sized corpora will also directly yield different frequencies for individual words. Thus, a good choice for the frequency cut-off is dependent on the underlying corpus and needs to be estimated carefully or learned from the data. As an alternative, a normalization scheme such as TF-IDF or relative frequencies could be investigated.

Clusters and Bigger Treebanks As the application of clusters helps with data sparseness, the question arises whether clusters still help if the size of the training data increases. In the best case, clusters would contribute to performance even for large amounts of training data. I cannot answer this question based on HeiST data as the treebank is too small. However, the Stanford Sentiment Treebank is large enough to perform this experiment.

I randomly sample 35000 reviews from the English DVD reviews provided by Blitzer et al. [2007]. For compatibility with the experiments performed on the German data, I create 1000 Brown clusters [Liang, 2005]. I keep all tokens irregardless of their frequency and truncate the cluster IDs to 15 significant bits. The clusters are applied to the Stanford Sentiment Treebank. The RNTN is then trained on increasing amounts of data and tested on the test set.

The results are shown in figure 4.9. It is quite clear that clusters are actually harmful to performance. A possible problem is that the clusters are created from a random sample of reviews. The German clusters are created from a domain-adapted subset of reviews (section 4.1.2). Still, at $n > 500$, the performance gap widens which indicates that larger treebanks do not benefit from the generalization effects of clusters.

4.3.4. Recovering lost sentiment information

Replacing words with their cluster IDs brings considerable performance improvements. However, the generalization step necessarily comes with a loss of information. In particular, the Brown clustering algorithm does not know about the semantic orientation of words. Words with different sentiment polarity can theoretically be allocated to the same cluster. During training, the RNTN sees the same token (cluster ID) with differing sentiment labels.

However, the sentiment information is easily recovered. Based on an external sentiment lexicon, the clusters are partitioned into subclusters based on their semantic orientation.

Method

The process is similar to the previously described cluster application step. The most likely cluster and the accompanying cluster ID for a word is looked up. Additionally, the word is looked up in a sentiment lexicon. If the word is negative, the cluster ID is affixed with **0**. For neutral or positive words, the affix is **1** respectively **2**.

I employ the SentiWS sentiment lexicon as it has been performed well in previous experiments.

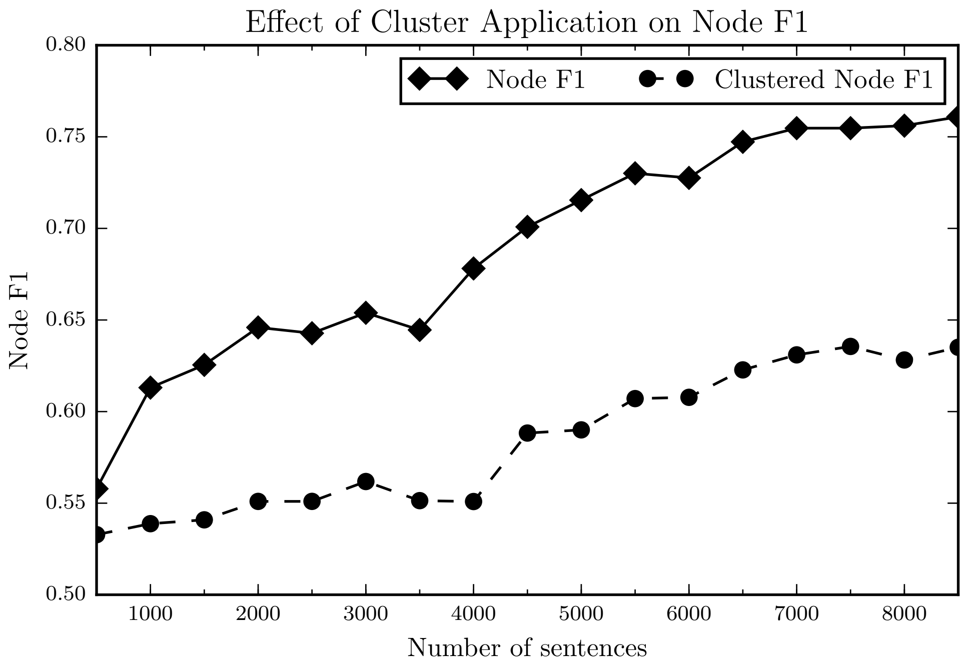


Figure 4.9.: Node F1 depending on the amount of unclustered and clustered data.

Evaluation

Application of divided clusters is evaluated on a sample of three good, mediocre and bad parameter sets as determined by node F1 measure. Additionally, the set of parameters is expanded with significant bits $[10, 11, 12, 13, 14]$ with their corresponding best frequency threshold. The additional parameters are included because the entire $[10; 14]$ range is missing from the initial set. Table 4.14 shows the results.

Ranking the parameter sets for plain and divided clusters (tables 4.13 and 4.14) and calculating the ranking agreement using Kendall's τ [Kendall, 1962] yields -0.326. Thus, a parameter set which yields good performance on undivided clusters does not necessarily carry over its advantage on divided clusters and is even likely to do badly. In fact, 8 or 9 significant bits perform worst on plain clusters and best on divided clusters.

Discussion

The cluster division method improves performance over the application of plain, undivided clusters. The cluster division process introduces more clusters. As seen for the plain clusters, an increase in the number of clusters is associated with an increase in performance. For this reason, I have to rule out the increase in cluster count alone as a reason for the performance increase.

Figures 4.10 and 4.11 show the influence of the number of clusters on the *affordable* data at c=5000. It is clear that for the same number of clusters, the divided clusters outperform the plain clusters.

Table 4.15 presents a summary of the systems described so far.

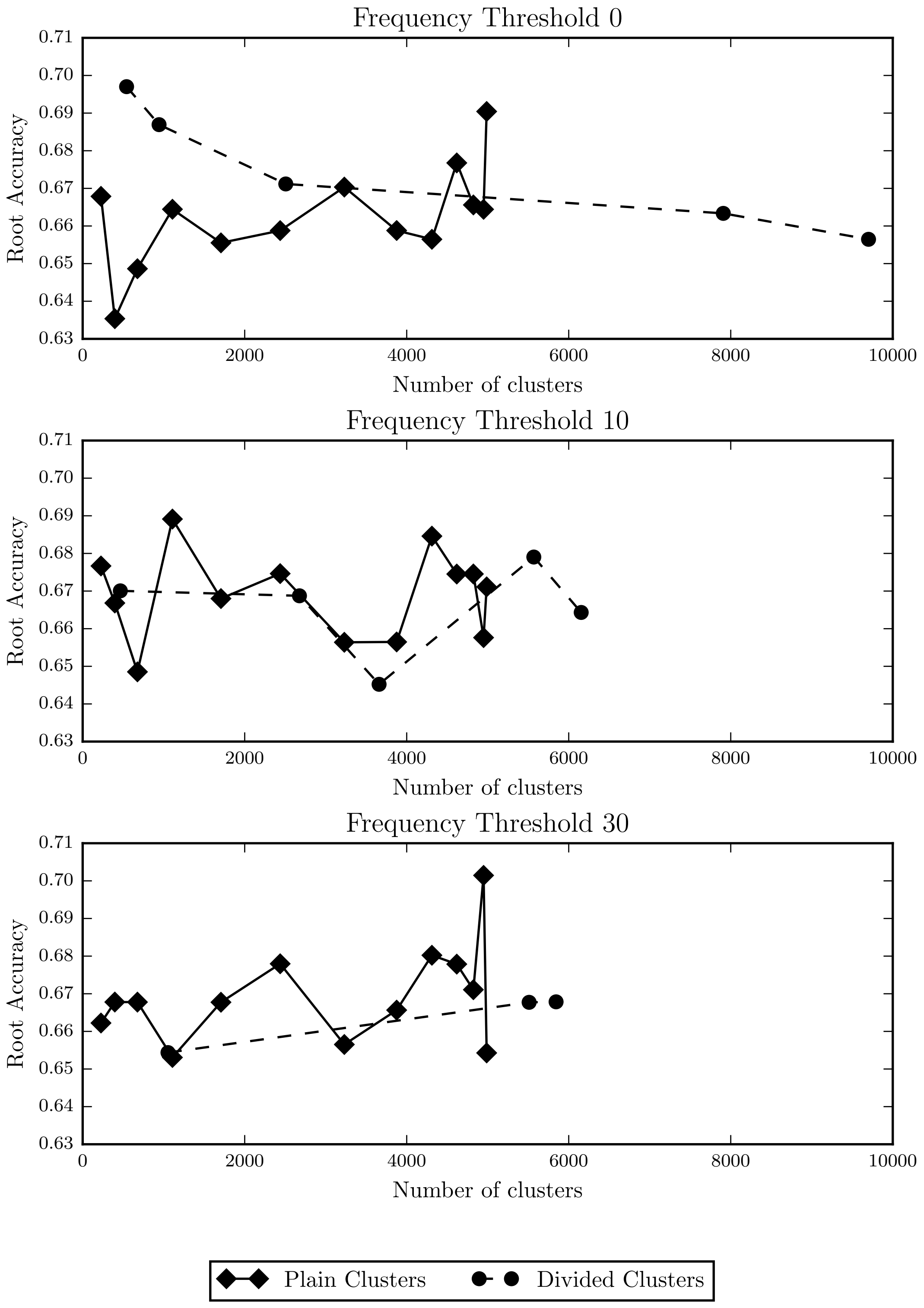


Figure 4.10.: Root accuracy at various cluster counts for plain and divided clusters.

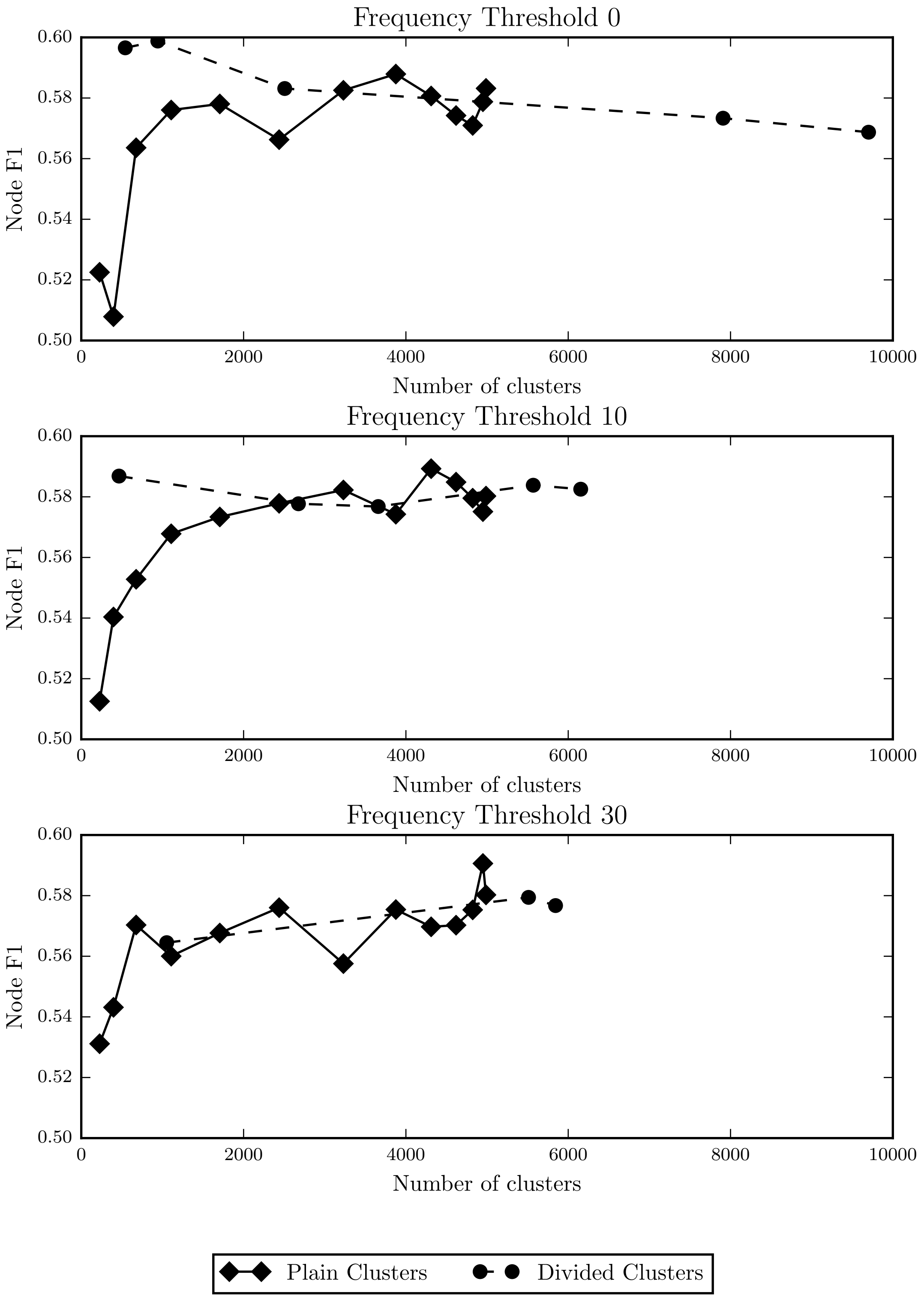


Figure 4.11.: Node F1 at various cluster counts for plain and divided clusters.

Freq Thresh	Significant Bits	Clusters	Node F1	Root Accuracy
0	8	226	0.523	0.668
	9	397	0.508	0.635
	10	676	0.564	0.649
	11	1108	0.576	0.664
	12	1707	0.578	0.656
	13	2438	0.566	0.659
	14	3230	0.583	0.670
	15	3876	**0.588**	0.659
	16	4311	0.581	0.656
	17	4618	0.574	0.677
	18	4823	0.571	0.666
	19	4949	0.579	0.664
	20	4987	0.583	**0.690**
10	8	226	0.513	0.677
	9	397	0.540	0.667
	10	676	0.553	0.649
	11	1108	0.568	**0.689**
	12	1707	0.573	0.668
	13	2438	0.578	0.675
	14	3230	0.582	0.656
	15	3876	0.574	0.656
	16	4311	**0.589**	0.685
	17	4618	0.585	0.675
	18	4823	0.580	0.675
	19	4949	0.575	0.658
	20	4987	0.580	0.671
30	8	226	0.531	0.662
	9	397	0.543	0.668
	10	676	0.570	0.668
	11	1108	0.560	0.653
	12	1707	0.568	0.668
	13	2438	0.576	0.678
	14	3230	0.558	0.657
	15	3876	0.575	0.666
	16	4311	0.570	0.680
	17	4618	0.570	0.678
	18	4823	0.575	0.671
	19	4949	**0.591**	**0.701**
	20	4987	0.580	0.654

Table 4.13.: Results for Amazon-based clusters (c=5000). Bold entries mark three winners per column.

Freq Threshold	Significant Bits	Clusters	Node F1	Root Acc
0	8	542	**0.597**	**0.697**
0	9	943	**0.599**	**0.687**
0	11	2507	0.583	0.671
0	15	7908	0.573	0.663
0	18	9701	0.569	0.656
10	8	465	**0.587**	0.670
10	12	2676	0.578	0.669
10	13	3659	0.577	0.645
10	15	5569	0.584	**0.679**
10	16	6154	0.583	0.664
30	10	1053	0.564	0.654
30	17	5511	0.579	0.668
30	19	5844	0.577	0.668

Table 4.14.: Results for the Amazon-Clusters (c=5000) divided by sentiment label. Bold marks overall winners per column.

System	Node F1	Root Accuracy
RNTN OPT + HeiST	0.590	0.679
RNTN Amazon Clusters (c=5000)	0.591	0.701
RNTN Clusters Divided (SentiWS)	0.599	0.687
Projection	0.591	0.751
Amazon Lasso	0.391	0.569
SentiWS	0.624	0.711

Table 4.15.: Performance for RNTN and other systems.

5. Combining the Components

5.1. Ensemble Learning

The individual components I present in the previous chapter show promising performance individually. A common method to combine the strengths of individual classifiers is ensemble learning. For example, the Projection method may contribute compositional information while the Amazon-based dictionary may add domain-specific vocabulary. See section 2.2.3 for an overview of related work on ensemble learning.

Several methods for ensemble learning exist. I evaluate several supervised classifiers and several combining of underlying base classifiers.

5.1.1. Features

The set of features is essentially the set of predictions made by the underlying classifier.

SentiWS

SentiWS contributes two features as described earlier. For a given phrase, two sums for positive and negative tokens are extracted. The sums are calculated over the sums for each token in the phrase as described in section 3.1.1.

Projection

The sentiment label as determined by the Projection method is extracted as a feature. The sentiment label is converted from the fine-grained scale to the coarse-grained scale.

Regression Score

The regressor trained on the Amazon *affordable* data set predicts a movie review rating for the given phrase. This feature is abbreviated as **RegScore** in the table. This is a variant of the rating-based feature described in Nguyen et al. [2014].

Amazon-Based Sentiment Lexicon

The phrase is classified into negative, neutral or positive according to the sentiment lexicon extracted from the Amazon movie reviews. This feature is abbreviated as **RegSenti** in the table.

Token Count

The token count is extracted as a feature. The number of leaves in a given parse tree is used as the token count.

POS Tags

The POS tags for the given phrase are extracted and the frequency of each individual POS tag is encoded. Thus, the POS extractor contributes more than just one or two dimensions to the resulting vector representation of the phrase. For each POS tag encountered, a new dimension is added. I use the Stanford POSTagger in version 3.0 [Toutanova et al., 2003] and the accompanying German model to obtain the POS tags. A tokenization step is not necessary as the individual tokens are extracted from the parse trees.

Note that the POS tagger sees the whole sentence. The phrase is a subtree of the parse tree for the full sentence. The sentence is tagged and the POS tags for the corresponding tree are extracted. This ensures that the POS tagger produces the best predictions possible. For implementation reasons, it is impractical to derive the POS tags directly from the parse trees.[1]

5.1.2. Learners

I select the following classes from the `sklearn.ensemble` package [Pedregosa et al., 2011].

- RandomForestClassifier
- AdaBoostClassifier
- GradientBoostingClassifier
- ExtraTreesClassifier

Additionally, I evaluate a standard SVM classifier with a linear kernel (`sklearn.svm.SVC` with parameter `kernel='linear'`). I include the SVM classifier to test whether a special ensemble classifier yields better performance in this setting.

These learners are used to learn the combination function for the base classifiers, i.e. the features described above. According to the taxonomy presented by Kittler [1998] (section 2.2.3), the ensemble learning setup described here uses **distinct representations** of the data, i.e. different features, to train each base classifier. The combination function for the base classifiers is then obtained by training a meta classifier.

Note that the learners for the combination function which come from the `sklearn.ensemble` package are ensemble learners themselves. This is a mere coincidence. These learners internally use decision trees. Decision trees support input of different types, e.g. continuous and ordinal features, which makes them a good fit for the task at hand.

[1]At time of feature extraction, the POS tags have been removed from the trees and only the basic skeleton remains along with the sentiment labels.

5.1.3. Initial Results - Learner and Feature Selection

The learners and feature sets are tested in a 10-fold cross-validation setting on the 20-sentences data set. Cross-validation splits are kept constant for all experiments.

Features	Classifier	F_{neg}	F_{pos}	Node F1
SentiWS + RegScore + Proj + Pos + Count	GradientBoosting	0.759	0.812	0.838
SentiWS + RegScore + Proj + Pos	GradientBoosting	0.752	0.814	0.836
SentiWS + RegSenti + Proj + Pos	GradientBoosting	0.740	0.805	0.828
SentiWS + RegSenti + Proj + Pos	RandomForest	0.688	0.797	0.808
SentiWS + RegScore + Proj + Pos	ExtraTrees	0.659	0.814	0.804
SentiWS + RegScore + Proj + Pos + Count	RandomForest	0.648	0.810	0.801
SentiWS + RegScore + Proj + Pos	RandomForest	0.650	0.807	0.800
SentiWS + RegSenti + Proj + Pos	ExtraTrees	0.660	0.796	0.797
SentiWS + RegScore + Proj + Pos + Count	ExtraTrees	0.625	0.816	0.793
SentiWS + RegScore + Proj	RandomForest	0.582	0.740	0.749
SentiWS + RegScore + Proj	GradientBoosting	0.545	0.759	0.745
SentiWS + RegScore + Proj	ExtraTrees	0.577	0.731	0.745
SentiWS + RegScore	RandomForest	0.543	0.742	0.738
SentiWS + RegScore	GradientBoosting	0.511	0.749	0.730
SentiWS + RegScore	ExtraTrees	0.484	0.736	0.715
SentiWS	ExtraTrees	0.396	0.728	0.681
SentiWS	GradientBoosting	0.396	0.724	0.679
SentiWS	RandomForest	0.387	0.718	0.675

Table 5.1.: Influence of feature sets and learners on classification performance in ensemble setting. Table is sorted by Node F1 column.

Table 5.1 lists the results of the selection process for best learner and feature sets sorted by Node F1 measure. The results show that the combination of several features improves performance. The addition of RegScore improves SentiWS performance considerably. RegScore performs slightly better than RegSenti for two out of three classifiers. The difference between RegScore and RegSenti is less than 0.01 points.

The performance improvement for both RegScore and RegSenti shows the usefulness of additional domain-specific sentiment information.

Introducing the projected labels improves performance by 0.01 to 0.02, depending on the classifier. Providing the part of speech feature set to the classifier yields another performance boost of 0.05 points. Thus, the POS tags can be considered a very important feature. The token count feature provides only a minor improvement for two out of three classifiers and a loss of 0.01 points for the ExtraTreesClassifier.

Combining more information from different sources improves performance over baseline in an ensemble learning framework. Comparing the SentiWS subset to the full set of features for GradientBoostingClassifier, the improvement is 0.159 points or 23.4%.

Table 5.2 lists the results for the remaining classifiers, SVC and AdaBoost. For some

Features	Classifier	F_{neg}	F_{pos}	Node F1
SentiWS + RegScore + Proj + Pos + Count	AdaBoost	0.565	0.758	0.748
SentiWS + RegScore + Proj + Pos	AdaBoost	0.573	0.754	0.750
SentiWS + RegSenti + Proj + Pos	AdaBoost	0.587	0.738	0.748
SentiWS	AdaBoost	NaN	0.692	NaN
SentiWS + RegScore	AdaBoost	NaN	0.663	NaN
SentiWS + RegScore + Proj	AdaBoost	NaN	0.556	NaN
SentiWS + RegSenti + Proj + Pos	SVC	0.565	0.748	0.746
SentiWS + RegScore + Proj + Pos	SVC	0.559	0.737	0.739
SentiWS + RegScore + Proj + Pos + Count	SVC	0.559	0.737	0.739
SentiWS + RegScore + Proj	SVC	NaN	0.566	NaN
SentiWS + RegScore	SVC	NaN	0.565	NaN
SentiWS	SVC	NaN	0.563	NaN

Table 5.2.: Performance of remaining classifiers.

experiments, calculating the F_{neg} and thus the macro average F1 is not possible as either precision or recall for the given class are null. Hence, F_{pos} and F_{neg} are included to allow a comparison where the macro average F1 is not available. Overall, the SVC and AdaBoost classifiers perform poorly compared to the other learners. In particular, the SVC is not a good fit for the ensemble setting.

5.1.4. Evaluation on Heidelberg Sentiment Treebank

In the previous sections, I evaluate a set of parameters on a small development set. The best performers are likely to perform well on the Heidelberg sentiment treebank. From the available learners, the GradientBoosting classifier performs best. A full feature set consisting of SentiWS, the predicted movie review score (RegScore), the projected sentiment label, the POS tags and the token count performs best. Some variations of the feature sets, such as removing the token count feature or exchanging the review score for a predicted sentiment label based on the learned Amazon sentiment lexicon (RegSenti) does not influence considerably. For this reason, these variations are also considered in the evaluation.

The evaluation is performed on the previously described ten-fold cross-validation fold of HeiST. The ensemble learner is trained on the train set and tested on the test set. Numbers reported are averages over all ten folds. Table 5.3 shows the results.

5.1.5. Discussion

The results from the top three feature sets are quite close. The second performer on the development set performs slightly worse on HeiST. Thus, the RegSenti feature performs slightly better than RegScore. This is surprising as the predicted review score can carry more information. This raises the question whether adding related feature combinations

Run	Features	Node F1	Root Acc
1	SentiWS + RegScore + Proj + POS + Count	0.658	0.765
2	SentiWS + RegScore + Proj + POS	0.652	0.744
3	SentiWS + RegSenti + Proj + POS	0.654	0.753
4	SentiWS + RegSenti + Proj + POS + Count	0.657	0.751
5	SentiWS + RegSenti + RegScore + Proj + POS	0.652	0.743
6	SentiWS + RegSenti + RegScore + Proj + POS + Count	0.658	0.762

Table 5.3.: Evaluation of GradientBoosting classifier and top three feature sets on Heidelberg sentiment treebank.

yields increased performance. Results are shown in table 5.3 in the bottom section. Run 4 tests the combination of RegSenti and the Count feature. Run 5 tests the combination of RegSenti and RegScore. Although the features come the same source, the classifier might benefit from the different representation. Run 6 finally adds the count feature on top of RegSenti and RegScore.

The result show that run 4, i.e. RegSenti instead of RegScore, does not outperform run 1. Although run 5 does outperform run 2, it does not beat run 3. This shows that the combination of RegSenti + RegScore does not necessarily outperform RegSenti alone. Run 6 includes the Count feature and outperforms its counterpart, run 1, at the fourth decimal (not shown here). The difference likely is not significant.

In summary, there is no clear advantage of RegSenti over RegScore and there is no reason to include both features. The feature set suggested by the initial experiments shows the best performance within margins of errors.

Table 5.4 shows a comparison between ensemble the performance of the individual components.

System	Node F1	Root Accuracy
Ensemble (Run 1)	0.658	0.765
RNTN OPT + HeiST	0.590	0.679
RNTN Amazon Clusters (c=5000)	0.591	0.701
RNTN Clusters Divided (SentiWS)	0.599	0.687
Projection	0.591	0.751
Amazon Lasso	0.391	0.569
SentiWS	0.624	0.711

Table 5.4.: Performance of ensemble and other systems.

Ensemble is the best system presented so far. Measures for both node-level and sentence-level have improved. In particular, Ensemble is better than the already strongly performing SentiWS and Projection.

Feature Weights

Feature weights are provided by scikit-learn in a `feature_importances` member of the fitted classifier object. The feature importances are derived from the trees internally used by GradientBoosting classifier. In the decision trees used as base classifiers for the GradientBoosting, each node is associated with an input feature. Each node partitions its associated region into subregions based on the value of the feature. The input feature used to split a particular node is the one that gives the maximal estimated improvement (given the loss function) compared to not splitting the node at all [Hastie et al., 2009][2]. The sum of these improvements over all nodes in the tree where the given feature is used as splitting variable is then considered the relative importance. For the GradientBoosting classifier, which uses multiple base classifiers, the average over the relative improvements for the feature is taken. The averages for the feature importances are then scaled to 100 to give the feature weights in percent.

A more intuitive, albeit not technically correct, interpretation is given by the scikit-learn documentation[3]. The relative depth of a feature (node) in a decision tree is an indicator of its importance. Nodes close to the root of the tree are used to predict a larger portion of samples. Thus, higher nodes as determined by an average over all trees are more important.

Table 5.5 shows the weights assigned to the individual features by the GradientBoosting classifier. The values are averaged over all ten folds.

Run	RegSenti	Proj	RegScore	Count	SentiWS P	SentiWS N	POS
1	–	0.1026	0.0576	0.0919	0.1315	0.1203	0.4961
2	–	0.1045	0.0554	–	0.1397	0.1289	0.5715
3	0.0096	0.1049	–	–	0.1416	0.1401	0.6038
4	0.0073	0.1027	–	0.0959	0.1343	0.1342	0.5256
5	0.0000	0.1045	0.0554	–	0.1397	0.1289	0.5715
6	0.0000	0.1026	0.0576	0.0919	0.1315	0.1203	0.4961

Table 5.5.: Feature importances averaged over all splits for the GradientBoosting classifier.

For POS, the sum over the individual POS tags is used.

For SentiWS, the positive and negative lists contribute an equal amount at 12% to 14% each. The POS feature is a collection of many features and thus contributes much more than the previously described features. The following most important feature is Projection. As shown before, the projection method performs well and can thus contribute to the Ensemble. The count feature at 9% is also useful in determining the correct sentiment.

[2]Chapter 10

[3]`http://scikit-learn.org/0.14/modules/ensemble.html#random-forest-feature-importance`, accessed 29-10-2014

The RegScore feature contributes 5%. RegSenti, on the other hand, only contributes less than 1% even when RegScore is absent. This further indicates that RegScore is a more robust feature than RegSenti.

Comparing the contribution of POS for the top three performers, it is clear that the addition of reliable feature does not necessarily increase the performance by a huge amount. The same information might already be given by an existing feature. Consider the two top performers. The count feature contributes 10%, which is mainly subtracted from POS. Yet, the performance increase is minimal.

On definitions

Typical definitions of ensemble learning assume that the base classifiers directly output class predictions. As an example, consider the *voting* decision combination function given by Alpaydin [2004] [4]:

$$y_i = \sum_{j=1}^{L} w_j d_{ji}$$

Here, the vote y_i for a given class C_i is the weighted average (w_j) of the votes d_{ij} over all L classifiers for class C_i. From the definition, it is clear that the combination function work only works if all classifiers predict the same set of classes.

The methods in this section work differently. In initial experiments, the base learners were indeed giving direct predictions for the sentiment labels. In subsequent experiments, I find that going beyond the initial definition and adding more complex features improves performance.

In particular, I consider the POS features as *descriptors of input conditions* [Kittler, 1998] which allow the meta classifier to dynamically select the correct base classifier. In case of the regression-based features, RegScore also performs better than RegSenti.

In summary, although the setup described here is not exactly true to the formal definition, the spirit of ensemble learning is observed: combine multiple classifiers for superior performance.

Another way to look at the method described in this section is that of extensive feature engineering. If you take offense to the misuse of the ensemble definition, consider the experiments described from this different perspective.

5.2. Uptraining

Petrov et al. [2010] introduce the concept of uptraining where two learners are combined in a two-stage setting. The second learner is trained on the output of the first learner. The goal is to combine the best characteristics of both learners. In particular, the method reduces the need for labeled data if the first learner produces good predictions. Unlabeled data typically is abundant, and so it is in the movie review domain. In their question

[4]Equation 15.3, Chapter 15.2

parsing task, adding 10^6 unlabeled sentences (which are classified by the first learner) yields the same improvements as adding 2000 labeled sentences.

5.2.1. Uptraining and the Heidelberg Sentiment Treebank

I aim to take advantage of the uptraining method by labeling additional training data with the Ensemble learner and then training the RNTN on a combination of noisily labeled data and the Heidelberg sentiment treebank.

Data Acquisition and Method

The unlabelled data is again obtained from Amazon movie reviews. The **newreleases** data set is adapted to the Film-Rezensionen.de as described in section 4.1 and 1000 reviews are retained. To prevent undesired bias, three movies and their corresponding reviews which were used in for the regression-based feature are removed from the newreleases data.

The remaining 885 reviews (9682 sentences) are prepared for the Ensemble learner as described earlier. In particular, they are translated to English and sentiment labels are projected as described in section 4.2. I use the the complex alignment model and implicit root node alignment.

Evaluation

The proposed method is evaluated in ten-fold cross-validation setting using the splits for the Heidelberg sentiment treebank described before.

The Ensemble learner is trained on the train sets and then used to predict the labels for the full unlabeled review data set. The train sets are then concatenated with the now noisily labeled review data set according to different weights and the RNTN is trained on the concatenated data. I choose 0, 1 and 23 as weights. At weight 0, the RNTN is trained only on the noisily labeled data. At weight 1, HeiST (592 sentences) is added to the 9682 noisily labeled sentences. At weight 23, the Heidelberg sentiment treebank is copied 23 times and the number of sentences for manually and noisily labeled data is roughly equal.

I reuse the train set both for the Ensemble learner and the RNTN to eliminate any influence of the training data on the results. This puts the focus on the idea of uptraining, i.e. the combination of two learners with different strengths.

Discussion

Table 5.6 lists the evaluation results for the uptrained RNTN and the previous systems.

At HeiST weight 0, where only the noisily labeled data is used, the performance already is comparable to the RNTN trained directly on the HeiST. The node F1 measure is 0.17 points lower, possibly because the Ensemble cannot adequately capture compositional effects. Root level accuracy, on the other hand, shows a minor improvement by 0.008 points. When the HeiST is added (weight 1), performance improves considerably by

System	Node F1	Root Accuracy
Uptrained Weight 0	0.573	0.687
Uptrained Weight 1	0.590	0.716
Uptrained Weight 23	0.612	0.669
Ensemble (Run 1)	0.658	0.765
RNTN OPT + HeiST	0.590	0.679
RNTN Amazon Clusters (c=5000)	0.591	0.701
RNTN Clusters Divided (SentiWS)	0.599	0.687
Projection	0.591	0.751
Amazon Lasso	0.391	0.569
SentiWS	0.624	0.711

Table 5.6.: Performance of uptrained RNTN and other systems.

0.017 points for node F1 and 0.036 points for root accuracy. The increase in performance is unlikely due to pure data size effects. Recall the discussion of the effect of training set size on performance (section 4.3.2). The approximate improvement per 500 sentences ($\approx$ size of a test set in HeiST) is 0.0118 for node F1 and 0.01 points for root accuracy. The improvements here easily exceed these figures, especially considering that the size of the noisily labeled data approaches 10000 where the effects of size alone slow down (see figure 4.8).

At weight 23, the number of sentences for HeiST and noisily labeled data is approximately equal. The rationale for the repeated inclusion of the HeiST is to give more weight to the higher-quality data. In other words, at weight 1, the noisily labeled data might drown out the manually labeled HeiST because it constitutes 90% of the data. Performance for node F1 indeed increases by 0.022 points compared to weight 1. On the other hand, root accuracy suffers by 0.047 points and shows the worst performance of all three runs at 66.9%.

I cannot pinpoint an exact cause of this unexpected drop in performance. A possible explanation is that I change the distribution of the data too much by increasing the proportion of the data. However, the test data comes from the HeiST. Thus, increasing the portion of the HeiST should actually increase performance as train and test distributions become more similar.

Another possible explanation is that the bigger data set containing more varied, heterogeneous data from two sources contains more compositional sentiment effects. The choice of the **numHid** hyperparameter influences the word vector size and thus the number of interactions that can be modeled. Given that I improve, at the same time, the amount, the quality and the distribution of the data (bringing it closer to the test sets), the previous explanation seems likely. It also explains why the increase in the node-level F1 measure. The shorter phrases are less subject to compositional errors and can benefit from the improved data. Root accuracy, on the other hand, fails because the compositional interactions are not captured adequately. A separate estimation run for the hyperparameters on the development set is indicated.

5.2.2. Clustered Uptraining

Uptraining, like the clustered RNTN (section 4.3.3), improves performance by incorporating additional unlabeled data. The two methods are independent and easily combined. In this section, I evaluate whether the application of clusters in an uptraining setting improves performance further. Both methods can be seen as a way to supply additional lexical knowledge. The contribution of the ensemble method goes beyond lexical items, however. The SentiWS dictionary, the regression-based sentiment lexicon and the projection method contribute additional sentiment knowledge to the uptraining setting. Thus, one could assume that the lexical knowledge introduced by the Brown clusters is made obsolete by the uptraining method which already incorporates additional unlabeled data. However, the clusters are based on a larger corpus and thus contribute more lexical items The hypothesis for the following experiments is that the cluster application improves the performance of the uptraining method.

Method

The application of clusters in the uptraining setting is a straightforward combination of both methods. The training data for uptraining is prepared as described in section 5.2.1. The cluster application preprocessing step is performed as shown in section 4.3.3. The RNTN is then trained on the modified treebank.

Evaluation

The evaluation uses the standard ten-fold cross-validation splits of the Heidelberg sentiment treebank. As the training set is much bigger than before, I increase the maximum training time to four days to allow all 400 iterations to complete. I use three best performing cluster setups from earlier experiments in section 4.3.3 based on node F1 measure. I choose the uptraining data set with HeiST weight 23. Table 5.7 shows the results.

Discussion

The first three rows in table 5.7 list the results for the uptrained, clustered RNTN sorted by node F1. First, I compare these results to the non-uptrained, clustered results in table 4.13. For both node F1 and root accuracy, the ordering of the runs stays the same. Thus, the relative influence of the frequency threshold and the number of significant bits stays constant regardless of whether the data set has been enriched with uptraining data.

The absolute performance for node F1 does not improve. The baseline, *Uptrained Weight 23*, yields 0.612. The best clustered run only yields 0.599. Root accuracy improves considerably up to 0.706 compared to 0.669 for the baseline. However,the unclustered *Uptrained Weight 1* run in line 5 yields 0.716. Recall the discussion for the unclustered results in section 5.2.1. It is likely that the bad performance of *Uptrained Weight 23* is

Run	Freq Thresh	Sig Bits	Node F1	Root Accuracy
Uptrained Clustered	10	16	0.599	0.706
Uptrained Clustered	30	19	0.599	0.688
Uptrained Clustered	0	15	0.586	0.688
Uptrained Weight 0	–	–	0.573	0.687
Uptrained Weight 1	–	–	0.590	0.716
Uptrained Weight 23	–	–	0.612	0.669
Ensemble (Run 1)	–	–	0.658	0.765
RNTN	–	–	0.590	0.679
RNTN Amazon Clusters	–	–	0.591	0.701
RNTN Clusters Divided	–	–	0.599	0.687
Projection	–	–	0.591	0.751
Amazon Lasso	–	–	0.391	0.569
SentiWS	–	–	0.624	0.711

Table 5.7.: Performance of uptrained RNTN (HeiST weight 23) after cluster application (Amazon, c=5000).

due to bad hyperparameter selection. In this case, *Uptrained Clustered* would lose any advantage.

5.2.3. Clustered Uptraining: Recovering lost sentiment information

As shown in section 4.3.4, recovering lost sentiment information by partitioning the clusters based on semantic orientation improves performance. In previous experiments, I use the SentiWS dictionary to recover sentiment information. As seen in section 5.1, an ensemble learner with appropriate features easily outperforms SentiWS. For this reason, I introduce a new cluster division method based on the ensemble results. In the cluster division task, the goal is to classify a single token. In previous experiments, I primarily investigate performance on parse tree fragments. I still expect better performance for Ensemble over SentiWS in the clustered uptraining task.

In this section, I thus test two hypotheses. First, I attempt to show that divided clusters perform even better than undivided clusters as this has been established in previous experiments. Second, I attempt to show that dividing the clusters based on ensemble predictions performs better than SentiWS-based divisions.

Method

The ensemble partition method is based on the ensemble classification method described in 5.1. Two feature extractors are used, SentiWS and RegScore. Projection is not useful here as only single words are classified and thus, there is no need for a compositional method.

The GradientBoosting classifier is trained on a labelled corpus and then predicts a coarse-grained sentiment label. The label is appended to the cluster ID and the divided clusters are applied to the uptraining data as described in 4.3.4.

Evaluation

The maximum training time is increased to four days. The evaluation is based on the best uptraining weight. The best parameters from previous experiments on divided clusters are used (section 4.3.4). The classifier for the ensemble-based partition method is trained on the train set of each split. Results are shown in table 5.8.

Divison Method	Freq Thresh	Sign Bits	Node F1	Root Accuracy
SentiWS	0	8	0.606	0.718
SentiWS	0	9	0.600	0.710
SentiWS	10	8	0.600	0.723
Ensemble	0	8	0.566	0.696
Ensemble	0	9	0.577	0.718
Ensemble	10	8	0.575	0.712

Table 5.8.: Results for the RNTN trained on the HeiST with divided clusters (Amazon affordable, c=5000)

Discussion

The SentiWS partition method clearly outperforms the Ensemble on Node F1 by a large margin. For root accuracy, two out of three SentiWS runs are better than any Ensemble run. Clearly, the SentiWS method is better than Ensemble overall. The node-level measure probably benefits more because the shorter phrase suffer harder from sparsity problems. For longer sentences, the overall sentiment may be easily recovered from the rest of the sentence if some words are unknown.

Compared to the non-uptrained results in section 4.3.4, the system with frequency threshold and significant bits $(10, 8)$ remains the worst performer out of the three runs selected. $(0, 8)$ and $(0, 9)$ switch first and second places.

The best results of all systems tested so far are listed in table 5.9

Partitioned the clusters based on sentiment information further improves performance compared to the undivided clusters. Node F1 increases by 0.7 points and root accuracy by 1.2 points. Compared to plain uptrained run at weight 23, the partitioned clusters improves by 4.9 points for root accuracy. More importantly, the partitioned clusters show improvements in node f1 and root accuracy simultaneously instead of just for one measure at weights 1 and 23.

I have presented three methods to improve the performance of the RNTN for a given amount of training data: clusters, divided clusters, and uptraining. Although all three methods offer a partial solution to the sparsity problem, their contributions do not overlap

System	Node F1	Root Accuracy
Uptrained Clusters	0.599	0.706
Uptrained Clusters Partitioned	0.606	0.718
Uptrained Weight 1	0.590	0.716
Uptrained Weight 23	0.612	0.669
Ensemble (Run 1)	0.658	0.765
RNTN	0.590	0.679
RNTN Amazon Clusters	0.591	0.701
RNTN Clusters Partitioned	0.599	0.687
Projection	0.591	0.751
Amazon Lasso	0.391	0.569
SentiWS	0.624	0.711

Table 5.9.: Performance of uptrained RNTN and other systems.

completely. A combination of the individual methods yields the greatest gains. Comparing the uptrained partitioned run to the plain RNTN, Node F1 increases by 1.6 points (2.7%) and root accuracy increases by 3.9 points (5.7%).

System	p Node F1	p Root Acc
SentiWS (No Shift)	**0.001**	0.078
Amazon Lasso (No Shift)	**0.001**	**0.001**
Projection	0.732	**0.001**
RNTN Clusters	0.590	0.708
RNTN Clusters Partitioned	**0.001**	**0.019**
Uptrain (HeiST x23)	**0.001**	0.584
Uptrain (HeiST x23) Clusters	0.0689	0.113
Uptrain (HeiST x23) Clusters Partitioned	**0.001**	0.31
Ensemble	**0.001**	**0.001**

Table 5.10.: Significance levels estimated by approximate randomization testing. Bold marks significant entries at $p < 0.05$

Significance Testing Table 5.10 shows the significance levels of the differences between a given system and the RNTN trained on HeiST as baseline. Significance levels are estimated using the approximate randomization test [Riezler and Maxwell, 2005]. Following the arguments put forth by Søgaard et al. [2014], I report the actual p-values instead of a boolean "significant/not significant". I refrain from performing a Bonferroni correction as the experiments and underlying hypotheses are independent.

Table 5.11 shows the significance levels of a comparison between the application of clusters and partitioned clusters. In both the plain HeiST and the uptrained setting, the

System	p Node F1	p Root Accuracy
HeiST	**0.001**	**0.003**
Uptrain	**0.001**	0.553

Table 5.11.: Significance levels for plain clusters versus partitioned clusters.

application of clusters confers a statistically significant benefit.

5.3. Error Analysis

5.3.1. Accuracy at n

Beyond the overall system performance as measured by node F1 and root accuracy, I analyse the predictions on a deeper level to get a better understanding of the individual strength and weaknesses. In a first analysis, I consider the accuracy of the systems at various phrase lengths n. The phrases are directly derived from HeiST. Table 5.12 gives a summary of the following figures. The number of wins indicates how often a system performs better over all n compared to the plain RNTN trained on HeiST.

System	# wins	# losses	# ties
All Neutral	2	49	1
SentiWS	35	6	11
Projection	32	13	7
Clusters	25	20	7
Partitioned	16	27	9
Uptrain	37	7	8
Ensemble	37	4	11

Table 5.12.: Performance for Accuracy @ n.

Figure 5.1 compares the performance of the plain RNTN against an 'all neutral' baseline. The majority of shorter phrases, e.g. single words, is neutral. The proportion of subjective phrases increases with phrase length. The 'support' line indicates, on a log scale, the amount of phrases at a given length. Starting at length 35, there are typically less than 10 phrases per length n. The 'support' plot is omitted in subsequent figures as it always the same.

Figure 5.2 compares the plain RNTN against the SentiWS lexicon. At length 1, the RNTN outperforms SentiWS. Starting at length 4, SentiWS tends to outperform the RNTN. This finding is quite surprising as the RNTN should have an advantage for longer phrases, not for shorter phrases, given its compositional nature. A possible explanation for the advantage of SentiWS is the small amount of training data for the RNTN. SentiWS might simply have better coverage. Under this hypothesis, errors at

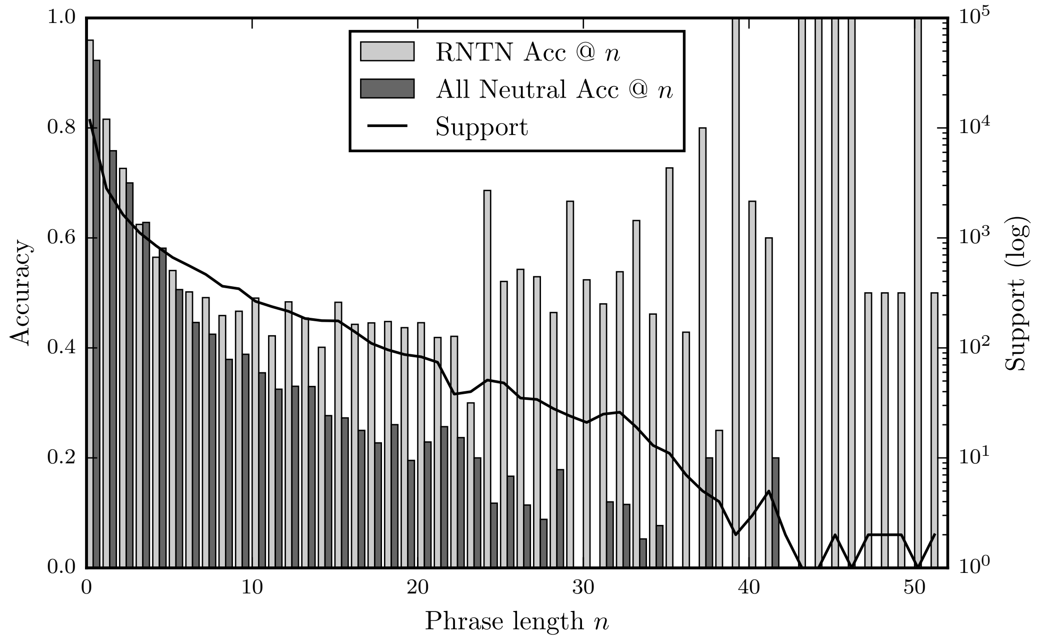


Figure 5.1.: Comparing RNTN to 'all neutral' baseline.

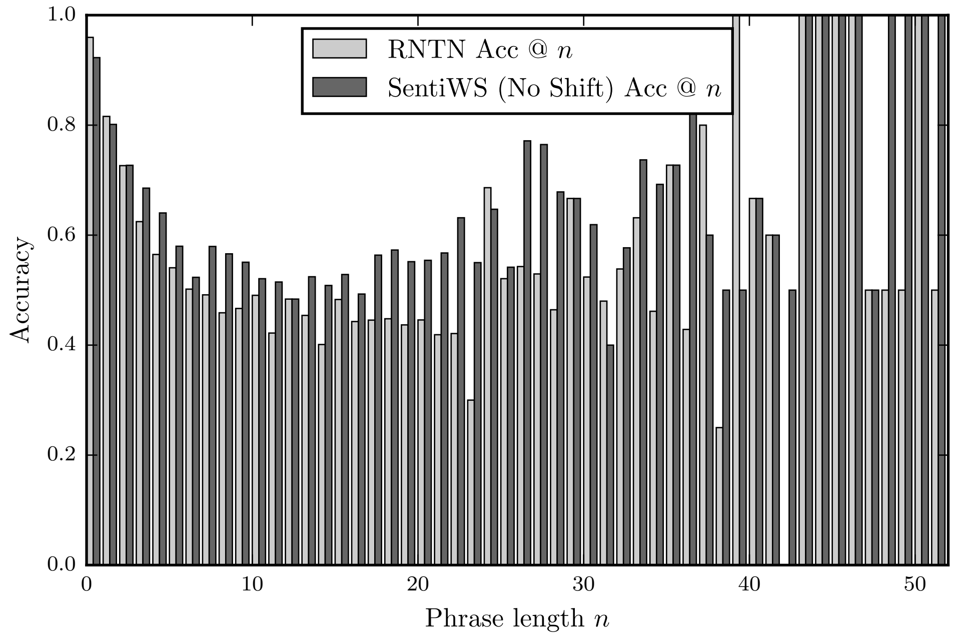


Figure 5.2.: Comparing RNTN to SentiWS.

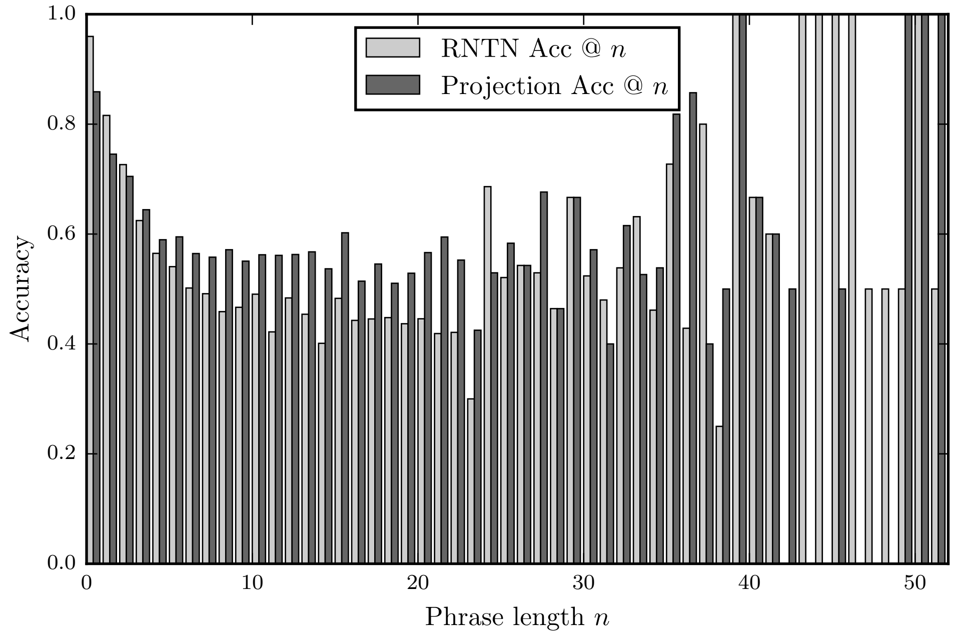


Figure 5.3.: Comparing RNTN to Projection.

$n < 4$ mostly arise when neutral items are missclassified as subjective. Further analysis is required here.

In figure 5.3, I compare the RNTN against projection. At $n < 3$, the RNTN outperforms projection. This is not surprising as the underlying `Lingua::Align` node alignment model likely works better for longer phrases. For longer phrases, projection outperforms the RNTN.

Figure 5.4 looks at the impact of the application of clusters on HeiST. For $n < 7$, the RNTN trained on plain HeiSt performs better by a small margin. A possible reason for this degradation is the loss of word-level information. For longer phrases, HeiST with clusters applied performs considerably better. Based on this observation, I hypothesize that clusters generalize well for the purpose of the compositionality function even if some loss on token level is observed.

In figure 5.5, I compare the accuracy of partitioned clusters against a plain RNTN. Unlike earlier figures, there is no clear value of n where one system performs better than the other. The partitioned clusters perform better for 16 values of n whereas the unchanged clusters outperform the plain RNTN in 25 cases. This discrepancy is surprising, given that the partitioned clusters perform much better than either plain or clustered HeiST in node F1 and root accuracy. However, accuracy at n gives bigger weight to the neutral class than the macro-average node F1, which may explain this discrepancy.

At $n = 1$, the partitioned clusters perform considerably worse than the untouched

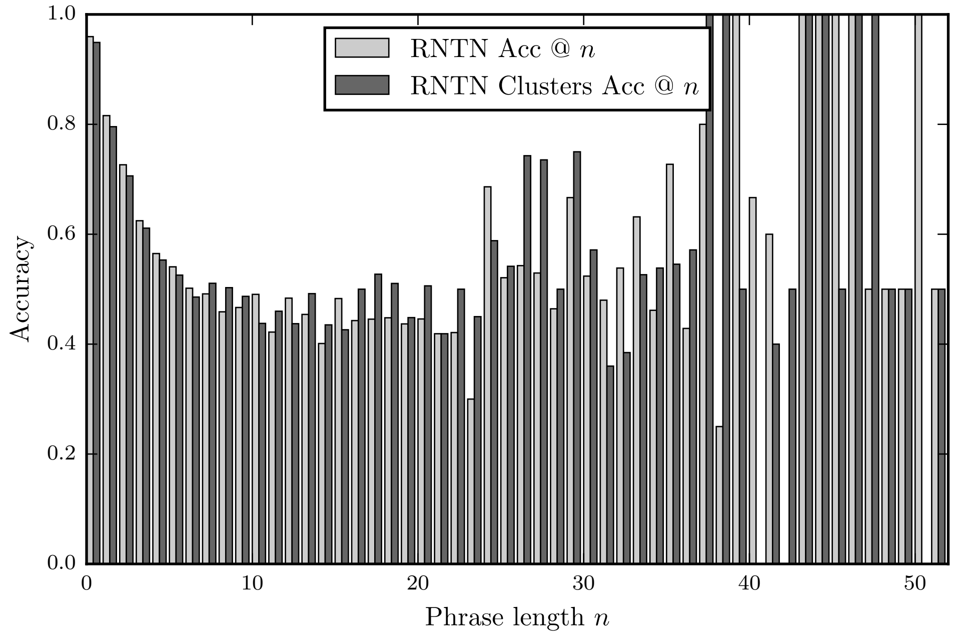


Figure 5.4.: Comparing RNTN to RNTN + Clusters.

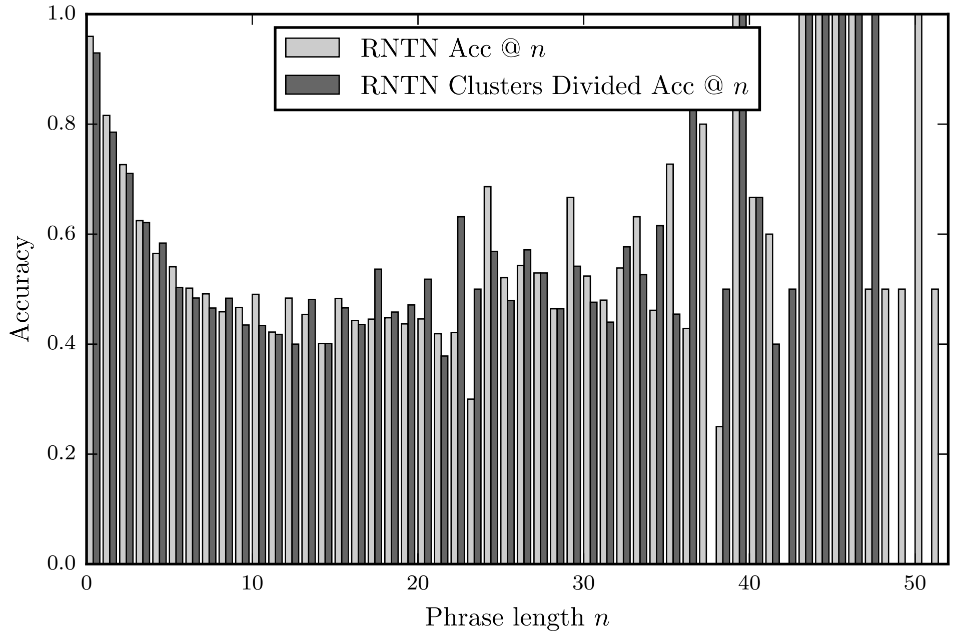


Figure 5.5.: Comparing RNTN to RNTN + Partitioned Clusters.

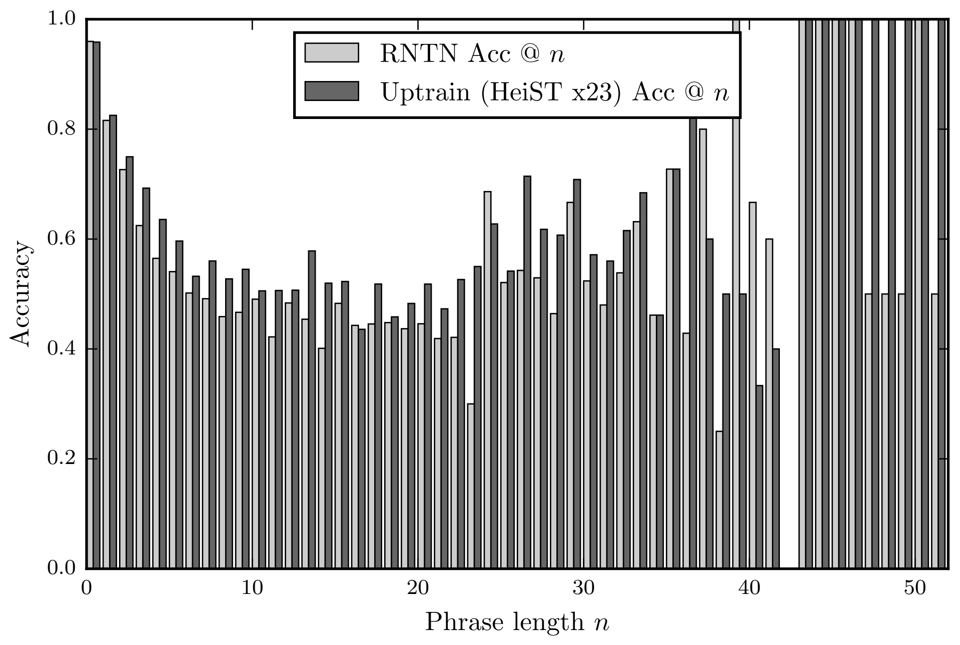


Figure 5.6.: Comparing RNTN to Uptrained RNTN.

clusters. This is surprising, considering that the partitioned clusters are much richer in sentiment information. The lower score is congruent with the value observed for pure SentiWS in figure 5.2. This indicates a mismatch between the prior polarity labels for individual words given by SentiWS and HeiST.

Figure 5.6 compares plain RNTN to the uptrained RNTN. Starting at $n > 1$, the uptrained system beats the plain RNTN. At the word level, the performance difference is negligible. The uptrained RNTN beats the plain RNTN in 37 cases, which is the best result of all comparisons in this section. The strong performance across short and long phrases indicates a contribution both on the lexical and the compositional level.

The ensemble system shown in table 5.7 also shows strong performance, congruent with the node F1 and root accuracy results. Similar to the uptrained RNTN, the ensemble outperforms the RNTN at phrases consisting of 2 or more words. The ensemble beats the RNTN in 37 cases, but the individual margins tend to be higher than for the uptrained RNTN.

5.3.2. Production-Level Accuracy

The HeiST is essentially a collection a binary parse trees where category labels have been replaced with sentiment labels. We can thus look at individual productions in a hypothetical sentiment grammar. Several of these productions realize linguistic phenomena

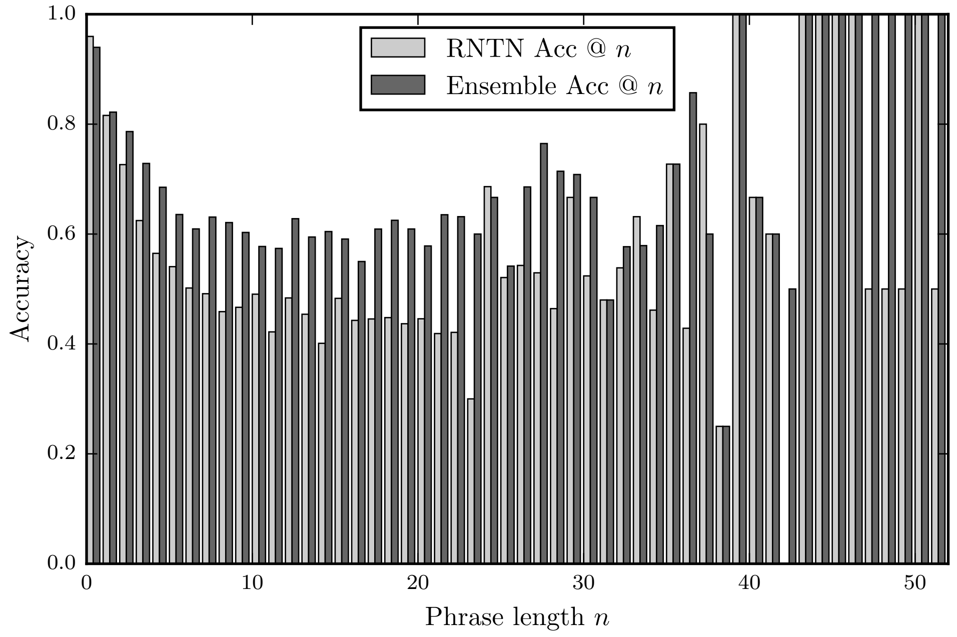


Figure 5.7.: Comparing RNTN to Ensemble.

related to the expression of sentiment such as negation and intensification[5]

Rules

INT Consider the phrase "sehr interessant" (*very interesting*), where "sehr" intensifies the adjective "interessant". In HeiST, the phrase is annotated as *(4 (2 sehr) (3 unterhaltsam))* and the corresponding production is [4 → 2 3]. Such productions, where the parent label is more intense than the average of its children, are categorized as **INT**. Intensification also works for negative sentiment: (0 (2 zu) (1 unglaubwürdig)) *(0 (2 too) (1 unbelievable)*.

INV Inversion, also known as polarity flipping, is realized by productions such as [1 *to* 2 3]. Consider "(1 (2 ohne) (3 (2 viel) (2 Tiefgang)))" *(1 (2 without) (3 (2 considerable) (2 substance)))*. These rules are marked as **INV**.

MWE Multi-word expressions are phrases where the sentiment of the full phrase is not immediately obvious given the sentiment of the constituents. The **MWE** rule includes subjective phrases with objective constituents and objective phrases with subjective constituents.

[5]The rules for these phenomena are the result of joint work between my supervisor Dr. Yannick Versley and I.

System	INV		ID		INT		MWE		AVG	
	Corr	Acc	Corr	Acc	Corr	Acc	Corr	Acc	Corr	Acc
SentiWS	31	0.223	148	**0.670**	200	**0.662**	273	**0.305**	1892	**0.551**
Proj	36	0.259	157	**0.710**	200	**0.662**	260	**0.291**	1901	**0.554**
RNTN	41	0.295	124	0.561	177	0.586	213	0.238	1591	0.463
Clusters	56	**0.403**	144	0.652	169	0.560	270	**0.302**	1638	0.477
Partitioned	49	**0.353**	142	0.643	174	0.576	237	0.265	1684	0.491
Uptrain	40	0.288	138	0.624	190	0.629	175	0.196	1638	0.477
Up Clust	43	**0.309**	145	0.656	175	0.579	185	0.207	1635	0.476
Up Part	31	0.223	133	0.602	160	0.530	130	0.145	1482	0.432
Ensemble	37	0.266	172	**0.778**	217	**0.719**	246	0.275	2027	**0.590**
Total per Rule	139		221		302		895		3433	

Table 5.13.: Accuracy for different rule types.

As examples, consider "(3 (2 viel) (2 Tiefgang))" (*(3 (2 considerable) (2 substance)*. Neutral phrases with subjective constituents are rather scarce in the data at 10 occurences and often represent noise, not true ambiguity: (2 (1 brutal) (1 (2 und) (2 dreckig))) *(2 (1 brutal) (1 (2 and) (2 dirty)))*.

The MWE phenomenon goes beyond commonly considered intensifiers and shifters and also poses a problem for sentiment lexicons which typically only consider single words.

AVG Averaging is another phenomenon found in the productions. If the sentiment of a parent node lies between or at the two extremes of its children, then the rule is considered **AVG**. As an example, consider [3 → 2 4] as found in "(3 (4 herrlich) (2 zynischen))" *(3 (4 blissfully) (2 zynical))*. This category also encompasses productions where the parent node is neutral or the same as one of the child nodes, e.g. "(2 (1 bösen) (2 Humors))" *(2 (1 dark) (2 humor))* and "(3 (3 Solide) (2 Unterhaltung))" *(3 (3 solid) (2 entertainment))*. The AVG group is the biggest group.

ID The **ID** rule subsumes all productions where parent and children share the same label, e.g. [3 → 3 3].

Rule Accuracy

Table 5.13 lists the accuracy in predicting the rule types for all systems presented. The accuracy is determined as the ratio of correct parent node labels to all parent node labels. Only positive and negative parent labels are considered and fine-grained labels are merged into coarse-grained labels. If a gold label is neutral, the rule instance is ignored for the purpose of accuracy computation. The motivation for this restriction is to highlight a system's ability to distinguish between positive and negative classes. As only parent labels are considered, the evaluation focusses on a system's ability to correctly model the phenomenon realized by the constituents.

The last row shows the total number of tree nodes matching a given rule, again ignoring neutral parent nodes.

For **INV**, the RNTN based systems perform best. Both plain RNTN and the uptrained RNTN perform comparatively bad and require clusters to improve accuracy considerably. Partitioning the clusters hurts performance, contrary to the results obtain in overall node f1 and root accuracy. Partitioned clusters in the uptrain settings perform as badly as SentiWS.

Ensemble, SentiWS and Projection perform best for the **ID** rules. These are the best performers overall. This is not surprising as there is not much compositional knowledge needed for rules of this type. On the other hand, these systems also perform best for **INT** and **AVG**. A simple explanation for this observation is that the best overall systems also perform well on the rules in question.

Finally, the best performers for **MWE** are SentiWS, Projection and the RNTN with clusters. It is surprising to see SentiWS as the best performer here as the lexicon only deals with single words. The generalization based on clusters apparently helps with multi-word expressions. Overall, accuracy for **INV** and **MWE** is low even for the best systems. For **INT**, the coarse-grained label space hides smaller nuances which may be modelled correctly by a system. More work is required to study and model these phenomena. In particular, measures which help overall performance hurt **INV**. This raises the question how relevant the linguistic phenomena listed here are overall for sentiment analysis. Negation handling is considered an important issue in sentiment analysis [Wiegand et al., 2010]. In my data, multi-word expressions outweigh inversion (i.e. negation) by far. Others have shown that bigram features, i.e. basic multi-word expressions, improve performance for sentiment analysis [Wang and Manning, 2012]. Thus, **MWE** are a promising phenomenon for future research

5.3.3. Contrastive Conjunctions

Socher et al. [2013] evaluate contrastive conjunctions where two phrases of opposing sentiment are joined by the conjunction "but". In a phrase "X but Y", the goal is to predict the correct sentiment of both X and Y and the full phrase.

For HeiST, I extract two kinds of phrases containing the conjunction "aber" (*but*). In the first kind, "aber" connects to main clauses to form a coordinated sentences. The second kind concerns coordinated adjective phrases where two adjective phrases are joined by "aber". See figures 5.8 and 5.9 for examples of both kinds of coordination.

I extract 36 sentences containing contrastive conjunctions using the Tregex tree query tool [Levy and Andrew, 2006]. All systems are evaluated on these sentences on their ability to predict the coarse sentiment label of both coordinated phrases and, separately, on the original Socher et al. [2013] task. In this stricter setting, the parent must be correctly predicted as well.

Table 5.14 lists the shows the accuracy values for all systems. The RNTN based systems outperform all other systems in this task. This indicates that the RNTN does

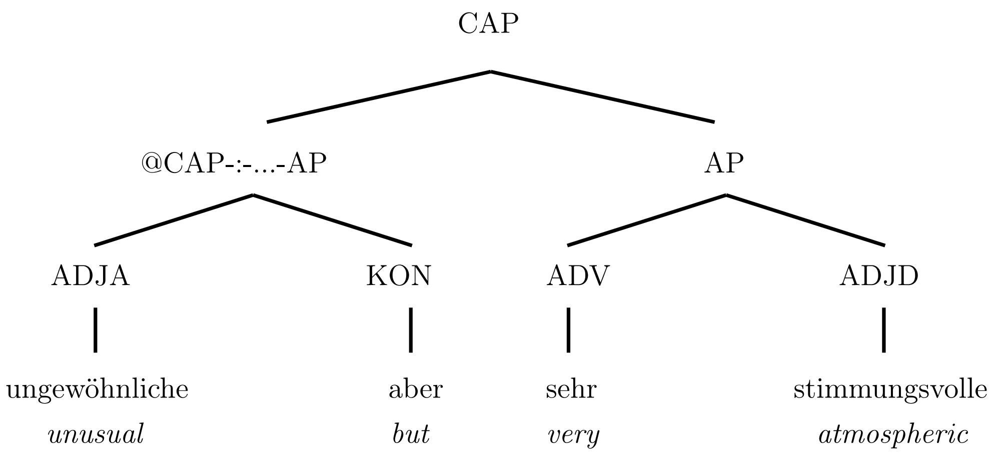


Figure 5.8.: Contrastive conjunction in coordinated adjective phrase.

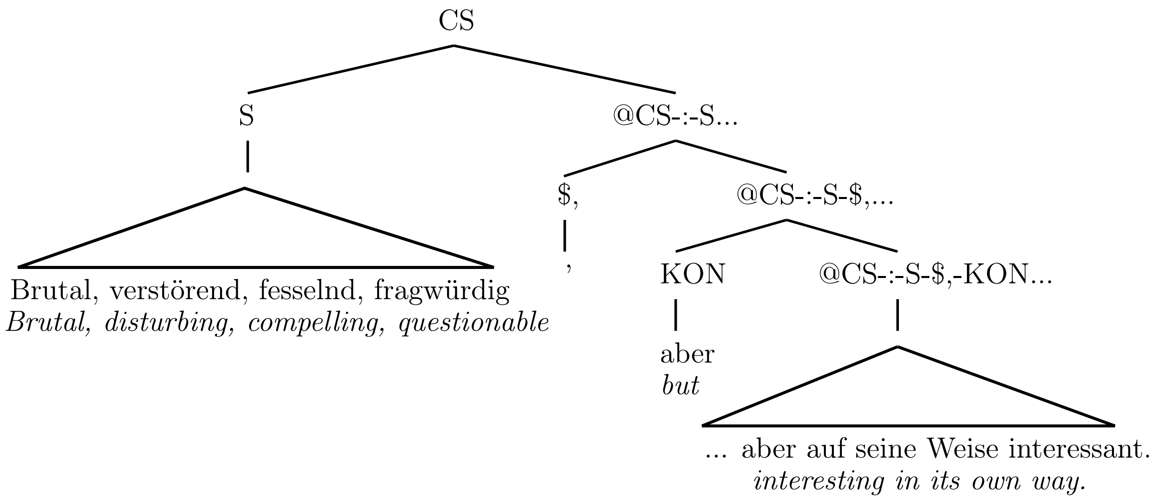

CS
S
@CS-:-S...
$,
@CS-:-S-$,...
KON
@CS-:-S-$,-KON...
Brutal, verstörend, fesselnd, fragwürdig
Brutal, disturbing, compelling, questionable
,
aber
but
... aber auf seine Weise interessant.
interesting in its own way.

Figure 5.9.: Contrastive conjunction in coordinated main clause.

indeed outperform more naive systems on compositionality-based task even when training data is scarce. The uptrain setting, albeit more accurate in node f1 and root accuracy, performs quite poorly among the RNTN systems. A possible explanation is that the training data generated by the ensemble is too noisy.

System	L+R Acc	P $\rightarrow$ L+R Acc
SentiWS (No Shift)	0.222	0.167
RNTN	0.306	0.278
RNTN Clusters	0.333	0.278
RNTN Clusters Partitioned	0.306	0.278
Uptrain (HeiST x23)	0.250	0.222
Uptrain (HeiST x23) Clusters	0.278	0.278
Uptrain (HeiST x23) Clusters Partitioned	0.278	0.222
Ensemble	0.111	0.083
Projection	0.139	0.111

Table 5.14.: Accuracy in the Contrastive Conjunctions Task.

6. Conclusion

6.1. Answers to Central Questions

In the introduction (section 2.1), I outline some central questions for my thesis. I reproduce the questions below for convenience.

6.1.1. Questions 1, 2 and 3: RNTN for German

I answer the first three questions together as the answers are closely related.

- How can I adapt the supervised Socher et al. [2013] method to German with as little monetary effort as possible?
- Do I have to create an expensive, manually annotated data set?
- In particular, how can I use the existing English data by Socher et al. [2013] for German sentiment analysis?

In light of the results I present, the cheapest method to use the RNTN in German is the projection method. The only cost arises from Google Translate, although a different machine translation system can be easily substituted. This is also the answer to question 2: no, a manually annotated data set is not strictly necessary. With regards to question 3: sentiment label projection via automatic parse tree alignment is a viable way to leverage the English Stanford sentiment treebank.

These results only apply for German, however. Few languages have access to manually aligned treebanks to train a tree aligner. In fact, creation of aligned treebanks requires trained annotators and a lot of work. Annotation of a sentiment treebank, on the other hand, is easily handled by untrained annotators. Although a sentiment treebank is more expensive, crowdsourced annotation is still very affordable. For $191, I actually obtain comparable performance on the subsentential level. A manually annotated treebank also enables more promising methods, such as ensemble or uptraining, which outperforms all other methods. Most importantly, a manually annotated sentiment treebank allows the proper evaluation of the final system.

Perspectives and Future Work It is possible, although not investigated here, to train the RNTN on the projected data, similar to the third experiment in Banea et al. [2008]. This setup also benefits from the improvements brought forth by cluster application.

6.1.2. Question 4: Improving Sentiment Analysis Results

- Can I increase system performance for a given data set with existing NLP tools and resources, without necessarily annotating more data?

Yes, it is possible to gain considerable performance improvements through standard NLP techniques. The cluster method requires an unlabeled corpus from which the clusters are learned. To recover lost sentiment information, a sentiment lexicon is necessary. If no off-the-shelf lexicon is available, then reviews with star ratings may be used as a weakly labeled corpus to extract a lexicon.

Perspectives and Future Work Another source of sentiment labels from which to recover lost information is the treebank data itself. Every label is annotated with the sentiment polarity for the attached word. It would be interesting to investigate whether a third party sentiment lexicon or the original treebank data produces superior performance. Another possible improvement comes from the clustering algorithm. Recovering lost sentiment information mainly becomes necessary because the clustering algorithm does not care about sentiment labels. A method to learn clusters which jointly considers prior sentiment polarity and word distribution might improve performance further. Lin et al. [2012] propose a joint sentiment-topic model which may serve as a starting point by interpreting the learned topics as word clusters.

Instead of resorting to a sentiment lexicon, such a clustering algorithm might also use star ratings as weak labels for the individual words. Domain-specific clusters perform better than generic clusters learned on newspaper corpora. Incorporating domain-specific sentiment information during clustering might convey similar benefits.

6.1.3. Question 5: Most useful methods regarding cost or performance

- Which methods are particularly useful regarding cost and/or performance?

If the best performance on the sentence level is desired, then the projection is the cheapest method. The best method, ensemble, only beats projection by a small margin.

Overall, the most “bang for your buck” is delivered by the cluster method. Using ubiquitous raw text as input and the Brown clustering algorithm, a simple preprocessing step yields considerable performance gains at no cost at all. Of course, this requires an annotated sentiment treebank, but this method provides the best value for (no) money. The other methods can be incorporated if better performance is required and the necessary data sets, e.g. sentiment lexicons, are freely available.

During annotation of the Heidelberg sentiment treebank, the subjectivity detection step provides substantial savings. Additionally, dividing the data set into smaller chunks and annotating the data in several steps helps. The annotation process can then be continuously refined and any errors can be rectified early. This is a considerable cost saver and should be considered a best practice. If I annotated the whole data set in one go, similar to a waterfall model in software engineering, much more items would have been faulty and due for re-annotation.

The ensemble method is also very useful in improving performance without annotating additional data. It is well-known that a meta-classifier improves classification accuracy (see section 2.2.3).

In languages where many sentiment analysis methods are available off-the-shelf, an ensemble classifier may be used to combine the individual strengths.

The previous considerations are only relevant if a reduction in annotation cost is desired. In a setting where engineer time is cheap and annotation is more expensive, it is necessary to get the most performance from a small data set. On the other hand, a freelancing software engineer bills about 70€ per hour[1] or about 500€ per work day. In a single day, a single engineer might produce minuscule performance gains and the money might be better spent annotating more data. However, this hypothetical software engineer might carefully consider the research questions and results described in this thesis. If they are applicable to the task at hand, considerable savings might be produced even on bigger data sets.

6.1.4. Question 6: Relationship between cost and performance

- Can I quantify the relationship between cost and performance?

Given the figures derived in section 4.3.2, I can estimate how much training data would be required to reach a given performance level. Based on the improvements shown by the individual methods, I calculate how many sentences would be needed for an equivalent increase in the evaluation measure. From the number of sentences saved, I can calculate the financial savings. The results are listed in tables tables 6.1 and 6.2. Although the projection method is not supervised and thus does not benefit from more training data, it is included in the table to give an estimate of its competetiveness.

System	Node F1	Increase	Saved #	Saved $
RNTN + HeiST	0.590	–	–	–
RNTN Clustered	0.591	*	–	–
RNTN Partitioned Clusters	0.599	0.009	450	$144
RNTN Uptrained (w=23)	0.612	0.022	1100	$352
RNTN Uptrained Clustered	0.599	*	–	–
RNTN Uptrained Partitioned Clusters	0.606	0.016	800	$256
Projection	0.591	*	–	—
Ensemble	0.658	0.68	3400	$1088

Table 6.1.: Improvements in node F1 compared to RNTN trained on HeiST. * indicates no statistically significant difference.

The figures presented here only present a very rough estimate as the underlying function is not entirely linear. The amount of money saved is calculated with an average price

[1]`https://www.gulp.de/knowledge-base/stundensaetze/stundensatz-forderung-der-it-engineering-freiberufler-nach-positionen%5B2%5D.html`, accessed 12-11-2014

System	Root Acc	Increase	Saved #	Saved $
RNTN + HeiST	0.679	–	–	–
RNTN Clustered	0.701	*	–	–
RNTN Partitioned Clusters	0.687	*	–	–
RNTN Uptrained (w=23)	0.669	*	–	–
RNTN Uptrained Clustered	0.706	*	–	–
RNTN Uptrained Partitioned Clusters	0.718	*	–	–
Projection	0.751	0.072	3050	$976
Ensemble	0.765	0.086	3644	$1166

Table 6.2.: Improvements in root accuracy compared to RNTN trained on HeiST. * indicates no statistically significant difference.

of $0.32[2] per sentence. Clearly, not all sentences are of equal length and frequent short phrases may already be annotated in larger corpora. Thus, the average price represents another simplification.

The improvements presented here are observed on a relatively small training corpus. As shown in section 4.3.3 with the SST data, the methods do not transfer easily to larger data sets where sparsity is less of an issue. Finding ways to employ cluster-based generalization or uptraining with larger corpora remains an interesting challenge.

6.2. Summary and Perspectives

In my thesis, I present the Heidelberg sentiment treebank, a new data set for compositional sentiment analysis in German. The data is downloaded from a German movie review blog and review summaries are extracted. Fine-grained sentiment polarity annotations are crowdsourced for sentence fragments derived from binary parse trees.

This data set enables new methods for compositional sentiment analysis in German. The sentiment polarity for a sentence can be derived recursively in a bottom-up fashion from its parts. The RNTN implements this method. I evaluate the RNTN on the Heidelberg sentiment treebank with promising results. Furthermore, I investigate the application of word clusters in the compositional sentiment task. The representation of words by their respective cluster IDs, thus abstracting away from individual words, improves classification performance considerably. In a novel extension of the cluster application approach, I recover lost sentiment information from a sentiment dictionary. By partitioning the clusters according to the prior sentiment polarity of their words, performance improves further.

A possible concern is the small size of the Heidelberg sentiment treebank compared to its english counterpart, the Stanford sentiment treebank, on which the RNTN is originally developed. To benefit from the vastly bigger Stanford corpus, I investigate the applicability of machine-translation based cross-lingual sentiment analysis methods.

[2]592 sentences in the HeiST, annotation costs $191

In the compositional setting, the projection of sentiment labels on the subsentential level poses a problem. I leverage automatic treebank alignment methods to successfully project the labels. In the evaluation, this novel cross-lingual sentiment analysis method outperforms the native RNTN trained on the smaller Heidelberg sentiment treebank. Thus, the proposed method successfully takes advantage of the bigger English sentiment treebank.

Additionally, I propose a novel method to extract sentiment lexicon from weakly labeled data. Product reviews with attached star ratings are ubiquitous. I rephrase the lexicon extraction problem as a regression learning task. A l1-regularized regression learner learns the most informative unigrams with regards to the star rating. The learned weights can be used directly as the semantic orientation value for the unigram. In comparison to SentiWS, an existing sentiment lexicon, performance in a sentiment analysis task is subpar.

The three methods I evaluate are very distinct, yet still aim to solve the problem of sentiment analysis. I combine the proposed methods in an ensemble learning setting to leverage their individual strengths. This combination of methods is very successful and outperforms the underlying systems easily.

Uptraining is another way to benefit from the strengths of different systems. A small, manually labeled corpus is combined with a larger, automatically labeled corpus. A classifier is trained on the data, with the expectation that it benefits from the noisily labeled data. Typically, the new classifier has some advantage over the classifier used to label the unannotated data. In the setup I propose, the ensemble learner predicts the labels for the raw data and the RNTN is trained on the noisily labeled data. The ensemble learner requires a machine translation system, German and English parsers, a German POS tagger and a RNTN with an English model to extract its features. The RNTN, uptrained once for German, is thus much faster at prediction time. The uptrained RNTN vastly outperforms the RNTN trained on the German Sentiment. Cluster-based optimizations also improve performance for the uptrained RNTN.

6.2.1. Perspectives

Beyond the possibilities for future work discussed above, syntax-backed compositional sentiment analysis may offer more exciting opportunities. Once interesting question is the applicability of the RNTN for aspect-based sentiment analysis. Existing work uses syntax-based patterns to extract entities, features and sentiment. The RNTN offers accurate sentiment analysis below the sentence level. Aspect-based methods may adapt their syntax patterns to the tree structures generated by the RNTN and benefit from increased accuracy.

My thesis is concerned with sentiment analysis on the movie review domain. Thus, reviews from other domains, such as cameras or books, are likely to achieve lower classification accuracy. Cross-domain sentiment classification remains an interesting task. The application of clusters, essentially a generalization method, has been shown to aid in crossing domains for flat, non-compositional sentiment analysis [Popat et al., 2013]. The applicability of this method in the compositional setting is an interesting research question. In this context, the generalization of the rating-based feature may also aid in cross-domain sentiment analysis. By applying clusters to weakly labeled review text, the rating-based feature may be adapted to a different domain. Standard domain adaptation techniques may additional be applied, e.g. instance selection as described in this work.

If the new domain has weakly labeled data available, then further adaptations may be pursued. As a presumed strength of the rating-based feature is its domain-specific knowledge, some effort must be made to remove features which hinder performance on the new domain. Gindl et al. [2010] describe a method to remove domain-dependent, i.e. harmful contexts. Essentially, they describe a method to solve the labeling mismatch problem. The resulting domain-independent rating-based feature is likely to be helpful for classifiers which have to handle several domains at once.

Going beyond the compositional vector-space approach, other methods for sentiment analysis have been proposed which handle subsentential units well. Hall et al. [2014] describe their SPAN parser which mostly relies on surface features with a minimal context-free grammar backbone. Features include unigrams, bigrams and POS tags. Their parser is easily adapted to the sentiment analysis task and outperforms the RNTN proposed by Socher et al. [2013]. Note that although the Hall et al. [2014] system is not strictly compositional, it relies on some tree-based features such as parent annotation. Considering the strong performance of less compositional methods like the SPAN parser or sentiment lexicons together with the surprisingly low incidence of compositional phenomena like intensification and polarity inversion (see section 5.3.2), the question arises if complex systems like the RNTN are indeed required. Multi-word expressions are a more common phenomenon and any improvements there are likely to benefit sentiment analysis as a whole.

A. Reproducing the Results

The ability to replicate results is a basic tenet of science. For the computer sciences, this entails the distribution of the program code and the data used to run the experiments. Although I describe the methods in sufficient detail to enable independent reproduction, the actual experiments are described in most detail in the program code. For this reason, I share the code and the data along with a set of scripts to run all experiments described in my thesis.

A.1. Basic Instructions

If you have not received a copy of the code and data along with my thesis, please download it from `https://github.com/mhaas/ma-thesis` or `http://www.computerlinguist.org/pages/ma-thesis/`. The archive contains a list of software dependencies required which is easily automatically installed. Run the script `thesis_run_ALL_experiments.sh` located in `code/scripts/` to run all experiments. Alternatively, choose one of the other scripts starting with `thesis_run` to run a single experiment. For convenience, some third-party data such as sentiment lexicons are included and clearly marked as such. Some data such as the token-level alignments required for the projection method are also included for convenience. A detailed set of instructions is listed in the `README.md`.

A.2. HeiST data

The Heidelberg sentiment treebank is included in the Penn treebank format. The annotations produced via Crowdflower are distributed in their original form. The IP addresses of the annotators are redacted. Additionally, the full set of translations of the Film-Rezensionen.de reviews is included as well. HeiST is released under the same Creative Commons BY-NC-SA license as the original reviews.

List of Figures

2.1. Recursive Neural Networks for Sentiment (Socher et al. [2013], Fig. 4) . . 18

3.1. Rating distribution for 631 Film-Rezensionen.de reviews. Y axis is log scaled. 38
3.2. Screenshot of the CrowdFlower task preview. 40
3.3. Example of default and deepened PP. 41

4.1. Comparison between random in-domain and out-of-domain documents (MSE). 57
4.2. Comparison between in-domain and out-of-domain documents (F1) 58
4.3. Comparison between most confident in-domain and random in-domain reviews. 58
4.4. Unaligned trees with sentiment labels on the English side. 61
4.5. Projecting the sentiment labels. 65
4.6. Testing root accuracy at various batch sizes and vector dimensions. 69
4.7. Testing node F1 at various batch sizes and vector dimensions. 70
4.8. Performance at various training set sizes. 71
4.9. Node F1 depending on the amount of unclustered and clustered data. . . 76
4.10. Root accuracy at various cluster counts for plain and divided clusters. . . 78
4.11. Node F1 at various cluster counts for plain and divided clusters. 79

5.1. Comparing RNTN to 'all neutral' baseline. 96
5.2. Comparing RNTN to SentiWS. 96
5.3. Comparing RNTN to Projection. 97
5.4. Comparing RNTN to RNTN + Clusters. 98
5.5. Comparing RNTN to RNTN + Partitioned Clusters. 98
5.6. Comparing RNTN to Uptrained RNTN. 99
5.7. Comparing RNTN to Ensemble. 100
5.8. Contrastive conjunction in coordinated adjective phrase. 103
5.9. Contrastive conjunction in coordinated main clause. 103

List of Tables

1.1. Ergebnisse auf Satzebene (Root Accuracy) und Phrasenebene (Node F1). */** kennzeichnet statistische Signifikanz $p < 0.05/p < 0.005$. 7

3.1. Subjectivity detection with GradientBoostingClassifier. 42
3.2. Most confident predictions for subjectivity detection with GradientBoostingClassifier. 42
3.3. Performance of different learners for SentiWS + Regression + POS @ 50%. Evalution performed on 20-sentences data set. 43
3.4. Examples of negative and positive noun phrases created from SentiWS. . . 44
3.5. The cross-product of determiners, adjectives and nouns forms the neutral noun phrases. 45
3.6. Fine-grained label distribution. 46
3.7. Coarse-grained label distribution. 46
3.8. Fine-grained sentence label distribution. 47
3.9. Coarse-grained sentence label distribution. 47
3.10. Contributor satisfaction on a five-point scale. Survey taken by 18 contributors. 48

4.1. Statistics for movie review corpora. 50
4.2. Influence of different text representations on Mean Squared Error. Bold entries mark the row winner. 54
4.3. Mean Squared Error of different feature sets across alphabet thresholds. . 54
4.4. Influence of learner selection on Mean Squared Error. 55
4.5. Number of features selected by each learner out of 9718 unigrams (n=1000, k=40000) for the last fold in 10-fold CV. 55
4.6. MSE when testing on the Film-Rezensionen.de data set. 56
4.7. Comparison between domain-specific learned sentiment lexicon and SentiWS. 59
4.8. Feature Sets for Lingua-Align 63
4.9. Evaluation results for Projection. 66
4.10. Performance for RNTN and other systems. 71
4.11. Results for TüBa-based clusters (c=1000). Bold entries mark three winners per column. 73
4.12. Results for Amazon-based clusters (c=1000). Bold entries mark three winners per column. 74
4.13. Results for Amazon-based clusters (c=5000). Bold entries mark three winners per column. 80

4.14. Results for the Amazon-Clusters (c=5000) divided by sentiment label. Bold marks overall winners per column. 81
4.15. Performance for RNTN and other systems. 81

5.1. Influence of feature sets and learners on classification performance in ensemble setting. Table is sorted by Node F1 column. 84
5.2. Performance of remaining classifiers. 85
5.3. Evaluation of GradientBoosting classifier and top three feature sets on Heidelberg sentiment treebank. 86
5.4. Performance of ensemble and other systems. 86
5.5. Feature importances averaged over all splits for the GradientBoosting classifier. 87
5.6. Performance of uptrained RNTN and other systems. 90
5.7. Performance of uptrained RNTN (HeiST weight 23) after cluster application (Amazon, c=5000). 92
5.8. Results for the RNTN trained on the HeiST with divided clusters (Amazon affordable, c=5000) 93
5.9. Performance of uptrained RNTN and other systems. 94
5.10. Significance levels estimated by approximate randomization testing. Bold marks significant entries at $p < 0.05$ 94
5.11. Significance levels for plain clusters versus partitioned clusters. 95
5.12. Performance for Accuracy @ n. 95
5.13. Accuracy for different rule types. 101
5.14. Accuracy in the Contrastive Conjunctions Task. 104

6.1. Improvements in node F1 compared to RNTN trained on HeiST. * indicates no statistically significant difference. 107
6.2. Improvements in root accuracy compared to RNTN trained on HeiST. * indicates no statistically significant difference. 108

Bibliography

Ethem Alpaydin. *Introduction to machine learning.* MIT press, 2004.

Malik Atalla, Christian Scheel, Ernesto William De Luca, and Sahin Albayrak. Investigating the applicability of current machine-learning based subjectivity detection algorithms on german texts. *ROBUS 2011*, page 17, 2011.

A. R. Balamurali, Mitesh M. Khapra, and Pushpak Bhattacharyya. Lost in translation: Viability of machine translation for cross language sentiment analysis. In *Proceedings of the 14th International Conference on Computational Linguistics and Intelligent Text Processing - Volume 2*, CICLing'13, pages 38–49, Berlin, Heidelberg, 2013. Springer-Verlag. ISBN 978-3-642-37255-1.

Carmen Banea, Rada Mihalcea, Janyce Wiebe, and Samer Hassan. Multilingual subjectivity analysis using machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 127–135. Association for Computational Linguistics, 2008.

Daniel M. Bikel and Jeffrey S. Sorensen. If we want your opinion. In *Proceedings of the First IEEE International Conference on Semantic Computing (ICSC 2007), September 17-19, 2007, Irvine, California, USA*, pages 493–500. IEEE Computer Society, 2007. doi: 10.1109/ICSC.2007.81. URL `http://dx.doi.org/10.1109/ICSC.2007.81`.

John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic.* The Association for Computational Linguistics, 2007. URL `http://aclweb.org/anthology-new/P/P07/P07-1056.pdf`.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, 1992.

Bo Chen, Hui He, and Jun Guo. Language feature mining for document subjectivity analysis. In *Proceedings of the First International Symposium on Data, Privacy, and E-Commerce (ISDPE) 2007.*, pages 62–67. IEEE, Nov 2007. doi: 10.1109/ISDPE.2007.105.

Yejin Choi and Claire Cardie. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference* *on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 793–801, Stroudsburg, PA, USA, 2008. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1613715.1613816`.

Fellbaum Christiane. *WordNet*. Blackwell Publishing Ltd, 2012. ISBN 9781405198431. doi: 10.1002/9781405198431.wbeal1285. URL `http://dx.doi.org/10.1002/9781405198431.wbeal1285`.

Simon Clematide and Manfred Klenner. Evaluation and extension of a polarity lexicon for german. In *Proceedings of the First Workshop on Computational Approaches to Subjectivity and Sentiment Analysis*, 2010.

Simon Clematide, Stefan Gindl, Manfred Klenner, Stefanos Petrakis, Robert Remus, Josef Ruppenhofer, Ulli Waltinger, and Michael Wiegand. MLSA - A multi-layered reference corpus for german sentiment analysis. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012), Istanbul, Turkey, May 23-25, 2012*, pages 3551–3556. European Language Resources Association (ELRA), 2012. URL `http://www.lrec-conf.org/proceedings/lrec2012/summaries/125.html`.

Kerstin Denecke. How to assess customer opinions beyond language barriers? In Pit Pichappan and Ajith Abraham, editors, *Third IEEE International Conference on Digital Information Management (ICDIM), November 13-16, 2008, London, UK, Proceedings*, pages 430–435. IEEE, 2008a. doi: 10.1109/ICDIM.2008.4746812. URL `http://dx.doi.org/10.1109/ICDIM.2008.4746812`.

Kerstin Denecke. Using sentiwordnet for multilingual sentiment analysis. In *Proceedings of the 24th International Conference on Data Engineering Workshops, ICDE 2008, April 7-12, 2008, Cancún, México*, pages 507–512. IEEE Computer Society, 2008b. doi: 10.1109/ICDEW.2008.4498370. URL `http://dx.doi.org/10.1109/ICDEW.2008.4498370`.

Xiaowen Ding, Bing Liu, and Philip S Yu. A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining*, pages 231–240. ACM, 2008.

Yan-Shi Dong and Ke-Song Han. A comparison of several ensemble methods for text categorization. In *2004 IEEE International Conference on Services Computing (SCC 2004), 15-18 September 2004, Shanghai, China*, pages 419–422. IEEE Computer Society, 2004. doi: 10.1109/SCC.2004.1358033. URL `http://doi.ieeecomputersociety.org/10.1109/SCC.2004.1358033`.

Kevin Duh, Akinori Fujino, and Masaaki Nagata. Is machine translation ripe for cross-lingual sentiment classification? In *The 49th Annual Meeting of the Association for*

Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA - Short Papers, pages 429–433. The Association for Computer Linguistics, 2011. URL `http://www.aclweb.org/anthology/P11-2075`.

Bradley Efron, Trevor Hastie, Iain Johnstone, Robert Tibshirani, et al. Least angle regression. *The Annals of Statistics*, 32(2):407–499, 2004.

Andrea Esuli and Fabrizio Sebastiani. Sentiwordnet: A publicly available lexical resource for opinion mining. In *Proceedings of LREC*, volume 6, pages 417–422, 2006.

Ronen Feldman. Techniques and applications for sentiment analysis. *Communications of the ACM*, 56(4):82–89, 2013.

Kuzman Ganchev, João Graça, and Ben Taskar. Better alignments = better translations? In Kathleen McKeown, Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *ACL 2008, Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics, June 15-20, 2008, Columbus, Ohio, USA*, pages 986–993. The Association for Computer Linguistics, 2008. URL `http://www.aclweb.org/anthology/P08-1112`.

Hatem Ghorbel and David Jacot. Sentiment analysis of french movie reviews. In *Advances in Distributed Agent-Based Retrieval Tools*, pages 97–108. Springer, 2011.

Stefan Gindl, Albert Weichselbraun, and Arno Scharl. Cross-domain contextualisation of sentiment lexicons. In *Proceedings of the 19th European Conference on Artificial Intelligence (ECAI 2010)*. European Coordinating Committee for Artificial Intelligence, 2010.

João Graça, Kuzman Ganchev, and Ben Taskar. Postcat-posterior constrained alignment toolkit. *The Prague Bulletin of Mathematical Linguistics*, 91:27–36, 2009.

David Leo Wright Hall, Greg Durrett, and Dan Klein. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, Volume 1: Long Papers*, pages 228–237. The Association for Computer Linguistics, 2014. URL `http://aclweb.org/anthology/P/P14/P14-1022.pdf`.

Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The elements of statistical learning*. Springer Series in Statistics. Springer, second edition, 2009.

Tin Kam Ho, Jonathan J. Hull, and Sargur N. Srihari. Decision combination in multiple classifier systems. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 16(1):66–75, 1994.

Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, KDD '04, pages 168–177, New York, NY, USA, 2004. ACM. ISBN 1-58113-888-1. doi: 10.1145/1014052.1014073. URL `http://doi.acm.org/10.1145/1014052.1014073`.

Maurice G. Kendall. *Rank Correlation Methods.* Charles Birchall & Sons ltd, London, 3 edition, 1962.

Josef Kittler. Combining classifiers: A theoretical framework. *Pattern analysis and Applications*, 1(1):18–27, 1998.

Dan Klein and Christopher D. Manning. Accurate unlexicalized parsing. In Erhard W. Hinrichs and Dan Roth, editors, *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan.*, pages 423–430. ACL, 2003. URL `http://acl.ldc.upenn.edu/acl2003/main/pdfs/Klein.pdf`.

Akshi Kumar and Teeja Mary Sebastian. Sentiment analysis: A perspective on its past, present and future. *International Journal of Intelligent Systems and Applications (IJISA)*, 4(10):1, 2012.

Roger Levy and Galen Andrew. Tregex and tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of the fifth international conference on Language Resources and Evaluation*, pages 2231–2234. LREC, 2006.

Shoushan Li, Chengqing Zong, and Xia Wang. Sentiment classification through combining classifiers with multiple feature sets. In *International Conference on Natural Language Processing and Knowledge Engineering, 2007. NLP-KE 2007*, pages 135–140. IEEE, 2007.

Percy Liang. Semi-supervised learning for natural language. Master's thesis, Massachusetts Institute of Technology, 2005.

Chenghua Lin, Yulan He, Richard Everson, and Stefan Ruger. Weakly supervised joint sentiment-topic detection from text. *IEEE Transactions on Knowledge and Data Engineering*, 24(6):1134–1145, 2012. ISSN 1041-4347. doi: http://doi.ieeecomputersociety.org/10.1109/TKDE.2011.48.

Bing Liu. Sentiment analysis and opinion mining. *Synthesis Lectures on Human Language Technologies*, 5(1):1–167, 2012.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In Dekang Lin, Yuji Matsumoto, and Rada Mihalcea, editors, *The 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, Proceedings of the Conference, 19-24 June, 2011, Portland, Oregon, USA*, pages 142–150. The Association for Computer Linguistics, 2011. URL `http://www.aclweb.org/anthology/P11-1015`.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014. URL `http://www.aclweb.org/anthology/P/P14/P14-5010`.

Andrew Kachites McCallum. Mallet: A machine learning for language toolkit. http://mallet.cs.umass.edu, 2002.

Rada Mihalcea, Carmen Banea, and Janyce Wiebe. Learning multilingual subjective language via cross-lingual projections. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, 2007.

Saif Mohammad, Cody Dunne, and Bonnie Dorr. Generating high-coverage semantic orientation lexicons from overtly marked words and a thesaurus. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 - Volume 2*, EMNLP '09, pages 599–608, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics. ISBN 978-1-932432-62-6. URL `http://dl.acm.org/citation.cfm?id=1699571.1699591`.

Quoc Dai Nguyen, Quoc Dat Nguyen, Thanh Vu, and Bao Son Pham. *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, chapter Sentiment Classification on Polarity Reviews: An Empirical Study Using Rating-based Features, pages 128–135. Association for Computational Linguistics, 2014. URL `http://aclweb.org/anthology/W14-2621`.

Bo Pang and Lillian Lee. A sentimental education: Sentiment analysis using subjectivity. In *Proceedings of ACL*, pages 271–278, 2004.

Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: 10.3115/1219840.1219855. URL `http://dx.doi.org/10.3115/1219840.1219855`.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1118693.1118704. URL `http://dx.doi.org/10.3115/1118693.1118704`.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

Slav Petrov, Pi-Chuan Chang, Michael Ringgaard, and Hiyan Alshawi. Uptraining for accurate deterministic question parsing. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 705–713. Association for Computational Linguistics, 2010.

Kashyap Popat, Balamurali A.R, Pushpak Bhattacharyya, and Gholamreza Haffari. The haves and the have-nots: Leveraging unlabelled corpora for sentiment analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 412–422, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P13-1041`.

Anna N Rafferty and Christopher D Manning. Parsing three german treebanks: lexicalized and unlexicalized baselines. In *Proceedings of the Workshop on Parsing German*, pages 40–46. Association for Computational Linguistics, 2008.

Robert Remus, Uwe Quasthoff, and Gerhard Heyer. Sentiws - A publicly available german-language resource for sentiment analysis. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010. URL `http://www.lrec-conf.org/proceedings/lrec2010/summaries/490.html`.

Stefan Riezler and John T Maxwell. On some pitfalls in automatic evaluation and significance testing for mt. In *Proceedings of the ACL workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 57–64, 2005.

Sven Rill, Sven Adolph, Johannes Drescher, Dirk Reinel, Jörg Scheidt, Oliver Schütz, Florian Wogenstein, Roberto V. Zicari, and Nikolaos Korfiatis. A phrase-based opinion list for the german language. In Jeremy Jancsary, editor, *11th Conference on Natural Language Processing, KONVENS 2012, Empirical Methods in Natural Language Processing, Vienna, Austria, September 19-21, 2012*, volume 5 of *Scientific series of the ÖGAI*, pages 305–313. ÖGAI, Wien, Österreich, 2012. URL `http://www.oegai.at/konvens2012/proceedings/46_rill12w/`. PATHOS 2012 workshop.

Sebastian Rudolph and Eugenie Giesbrecht. Compositional matrix-space models of language. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 907–916, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL `http://dl.acm.org/citation.cfm?id=1858681.1858774`.

Yvonne Samuelsson and Martin Volk. Automatic node insertion for treebank deepening. In *Proceedings of the Third Workshop on Treebanks and Linguistic Theories (TLT), Tübingen, Germany*, pages 127–136, 2004.

Yvonne Samuelsson and Martin Volk. Phrase alignment in parallel treebanks. In *Proceedings of the Fifth Workshop on Treebanks and Linguistic Theories (TLT), Prague, Czech Republic*, pages 91–102, 2006.

Kiran Sarvabhotla, Prasad Pingali, and Vasudeva Varma. Sentiment classification: a lexical similarity based approach for extracting subjectivity in documents. *Information Retrieval*, 14(3):337–353, 2011.

Christian Scheible and Hinrich Schütze. Sentiment relevance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 954–963. The Association for Computer Linguistics, 2013. URL `http://aclweb.org/anthology/P/P13/P13-1094.pdf`.

Thomas Scholz and Stefan Conrad. Opinion mining in newspaper articles by entropy-based word connections. In *EMNLP*, pages 1828–1839, 2013.

Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1631–1642, 2013.

Marina Sokolova and Guy Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

Anders Søgaard, Anders Johannsen, Barbara Plank, Dirk Hovy, and Hector Martinez. What's in a p-value in nlp. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning*. SIGNLL, 2014.

Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Computational linguistics*, 37(2):267–307, 2011.

Huifeng Tang, Songbo Tan, and Xueqi Cheng. A survey on sentiment detection of reviews. *Expert Systems with Applications*, 36(7):10760–10773, 2009.

O. Tange. Gnu parallel - the command-line power tool. *;login: The USENIX Magazine*, 36(1):42–47, Feb 2011. URL `http://www.gnu.org/s/parallel`.

Heike Telljohann, Erhard W. Hinrichs, and Sandra Kübler. The tüba-d/z treebank: Annotating german with a context-free backbone. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association, 2004. URL `http://www.lrec-conf.org/proceedings/lrec2004/pdf/135.pdf`.

Jörg Tiedemann. Lingua-align: An experimental toolbox for automatic tree-to-tree alignment. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the*

International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta. European Language Resources Association, 2010. URL `http://www.lrec-conf.org/proceedings/lrec2010/summaries/144.html`.

Jörg Tiedemann and Gideon Kotzé. Building a large machine-aligned parallel treebank. In *Proceedings of the 8th International Workshop on Treebanks and Linguistic Theories (TLT'08)*, pages 197–208. EDUCatt, 2009.

Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

Peter D. Turney. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073153. URL `http://dx.doi.org/10.3115/1073083.1073153`.

Martin Volk, Anne Göhring, Torsten Marek, and Yvonne Samuelsson. SMULTRON (version 3.0) — The Stockholm MULtilingual parallel TReebank. http://www.cl.uzh.ch/research/paralleltreebanks_en.html, 2010. An English-French-German-Spanish-Swedish parallel Treebank with sub-sentential alignments.

Wiebke Wagner. Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit - o'reilly media, beijing, 2009, ISBN 978-0-596-51649-9. *Language Resources and Evaluation*, 44(4):421–424, 2010. doi: 10.1007/s10579-010-9124-x. URL `http://dx.doi.org/10.1007/s10579-010-9124-x`.

Ulli Waltinger. Germanpolarityclues: A lexical resource for german sentiment analysis. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010. URL `http://www.lrec-conf.org/proceedings/lrec2010/summaries/91.html`.

Sida Wang and Christopher D Manning. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 90–94. Association for Computational Linguistics, 2012.

Matthew Whitehead and Larry Yaeger. Sentiment mining using ensemble classification models. In *Innovations and Advances in Computer Sciences and Engineering*, pages 509–514. Springer, 2010.

Casey Whitelaw, Navendu Garg, and Shlomo Argamon. Using appraisal groups for sentiment analysis. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, pages 625–631. ACM, 2005.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2-3):165–210, 2005.

Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. 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. Association for Computational Linguistics, 2010.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the conference on human language technology and empirical methods in natural language processing*, pages 347–354. Association for Computational Linguistics, 2005.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational linguistics*, 35(3):399–433, 2009.

Rui Xia, Chengqing Zong, and Shoushan Li. Ensemble of feature sets and classification algorithms for sentiment classification. *Information Sciences*, 181(6):1138–1152, 2011.

Rui Xia, Chengqing Zong, Xuelei Hu, and Erik Cambria. Feature ensemble plus sample selection: Domain adaptation for sentiment classification. *IEEE Intelligent Systems*, 28 (3):10–18, 2013. doi: 10.1109/MIS.2013.27. URL `http://doi.ieeecomputersociety.org/10.1109/MIS.2013.27`.