

Structurally informed methods for improved sentiment analysis

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

Vorgelegt von
Stefanie Wiltrud Kessler
aus Stuttgart

Hauptberichter Prof. Dr. Jonas Kuhn
Mitberichter Prof. Dr. Manfred Stede

Tag der mündlichen Prüfung: 22. November 2016

Institut für Maschinelle Sprachverarbeitung
der Universität Stuttgart

2017

Erklärung (Statement of Authorship)

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbständig verfasst habe und dabei keine andere als die angegebene Literatur verwendet habe. Alle Zitate und sinngemäßen Entlehnungen sind als solche unter genauer Angabe der Quelle gekennzeichnet.

I hereby declare that the text at hand is the result of my own work and that I have not used sources without referencing them in the text. Any thoughts from others or literal quotations are clearly marked.

(Stefanie Wiltrud Kessler)

Contents

1. Introduction	1
1.1. Motivation	1
1.2. Research questions and contributions	3
1.3. Structure of this thesis	7
1.4. Publications	9
2. Sentiment Analysis	11
2.1. Introduction	11
2.2. Basic concepts in sentiment analysis	14
2.2.1. Subjectivity and sentiment relevance	14
2.2.2. Sentiment polarity and strength	15
2.2.3. Polarity clues and sentiment dictionaries	15
2.2.4. Prior and contextual polarity	16
2.2.5. Sentiment targets and aspect-oriented sentiment analysis	17
2.3. Structurally informed sentiment analysis	19
2.3.1. Structure in rule-based approaches	19
2.3.2. Structural features for machine learning	20
2.3.3. Machine learning guided by syntax trees	22
2.4. Negation and polarity reversers	23
2.4.1. Negation identification	23
2.4.2. Negation scope detection	25
2.4.3. Negation processing	26
2.4.4. Related phenomena	29
2.5. Comparisons in product reviews	31
2.5.1. Comparison sentence identification	32
2.5.2. Comparison component detection	34
2.5.3. Identification of comparison type and direction	35
2.6. Summary	38
3. Polarity reversing constructions	39
3.1. Introduction	39
3.2. Methods	42
3.2.1. Polarity and consistency classification	43
3.2.2. Representation of context as syntactic constructions	44
3.2.3. Generation of training examples	46
3.2.4. Extraction of polarity reversing constructions	47
3.2.5. Filtering of generated training examples	49
3.3. Automatic polarity annotation of sentences	53

3.4.	Experiments on PRC extraction	57
3.4.1.	Data and experimental setup	57
3.4.2.	Results and discussion	59
3.5.	Experiments on consistency and polarity classification	64
3.5.1.	Data and experimental setup	64
3.5.2.	Results and discussion	67
3.6.	Summary	71
4.	A corpus of comparisons in product reviews	73
4.1.	Introduction	73
4.2.	Existing data sets	74
4.3.	Data sources and annotation procedure	76
4.4.	Annotation scheme	79
4.5.	Analysis of the data	85
4.5.1.	Inter-annotator agreement	85
4.5.2.	Statistics and comparison to other data sets	88
4.5.3.	Discussion	91
4.6.	Summary	95
5.	Detecting comparisons in product reviews	97
5.1.	Introduction	97
5.2.	Approach	101
5.2.1.	Using an SRL approach to detect comparisons	101
5.2.2.	Sentence context	102
5.2.3.	Context from knowledge about the task	103
5.2.4.	Context from annotation design	104
5.3.	Experiments	107
5.3.1.	Data and experimental setup	107
5.3.2.	Results	108
5.3.3.	Discussion	117
5.4.	Summary	120
6.	Semi-supervised training set expansion with structural alignment	121
6.1.	Introduction	121
6.2.	Semi-supervised learning and related work	124
6.3.	Structural alignment approach	126
6.3.1.	Outline of structural alignment	126
6.3.2.	Sentence selection	129
6.3.3.	Creation of argument candidates	129
6.3.4.	Argument similarity scores	133
6.3.5.	Predicate similarity scores	136
6.3.6.	Alignments between candidate sets	137
6.3.7.	Label projection	139
6.4.	Analysis of expansion on development set	139
6.4.1.	Data and experimental setup	139

6.4.2. Results and discussion	142
6.5. Experiments on training set expansion	148
6.5.1. Data and experimental setup	148
6.5.2. Results	150
6.5.3. Discussion	154
6.6. Summary	156
7. Towards the prediction of product rankings based on comparisons	157
7.1. Introduction	157
7.2. Related work	159
7.3. Gold-standard rankings of products	162
7.4. Prediction of product rankings	168
7.4.1. Ranking methods	169
7.4.2. Aspect-specific rankings	171
7.5. Experiments	172
7.5.1. Data and experimental setup	172
7.5.2. Results and discussion	173
7.6. Summary	176
8. Conclusions and outlook	177
8.1. Summary of contributions	177
8.2. Possible directions for future work	179
A. Detailed results for Chapter 3	183
B. Detailed results for Chapter 5	189
C. Detailed results for Chapter 6	195
Bibliography	205

List of Figures

2.1. Reviews and star ratings	12
3.1. Dependency parse and syntactic constructions for an example	45
3.2. Steps for PRC extraction	48
3.3. Patterns used for aspect bigram extraction	51
3.4. Semi-structured review from <code>epinions.com</code>	54
3.5. Screenshot of the annotation tool for PRC annotation	58
3.6. Percentage of correctly extracted PRCs for base settings	60
3.7. Top 10 extracted PRCs with different scoring methods	61
3.8. Percentage of correctly extracted PRCs with filters	63
3.9. Results for consistency and polarity classification	68
4.1. Screenshot of the AMT annotation interface	78
4.2. Example for the fine-grained annotation of comparisons in a sentence	80
5.1. Steps of the comparison detection pipeline for an example sentence	102
5.2. Sentence context around an argument candidate example	103
5.3. Results of the experiments on sentence context	110
5.4. Results of the experiments on context from task and domain	113
5.5. Results of the experiments on multiword predicate annotations	115
5.6. Results of the experiments on argument annotations	116
6.1. Outline of structural alignment in pseudo-code	127
6.2. Argument candidate extraction for an example	130
6.3. Alignments and similarities for an example	138
6.4. Results on development set for labeled arguments	143
6.5. Results on development set for argument candidate extraction	145
6.6. Results on development set for predicate similarities	147
6.7. Results for argument identification on test set varying d and k	151
6.8. More results for argument identification for KK-CONTEXT-PREDS	153
6.9. Learning curves for argument identification	155
7.1. Screenshot of the top products in “DSRL cameras” from <code>amazon.com</code>	164
7.2. Screenshot of the product details for Canon EOS 1D X from <code>amazon.com</code>	165
7.3. Screenshot of the scores for Canon EOS 1D X from <code>snapsort.com</code>	166
7.4. Illustration of ranking based on comparisons	170

List of Tables

3.1. Statistics of automatically annotated PROSCONS data set	55
3.2. Agreement between manual and automatic annotation	55
3.3. Statistics of customer review data sets	66
4.1. Inter-annotator agreement on sentences annotated by all annotators . . .	86
4.2. Statistics about our data and other corpora	89
5.1. Statistics about the multiword predicates in the data	114
5.2. Statistics about the number of predicate annotations in the data	119
6.1. Overview of word similarity measures	136
6.2. Inter-annotator agreement on the development set	141
7.1. Comparison of the characteristics of different external gold rankings . . .	168
7.2. List of aspect phrases used for aspect-specific rankings	172
7.3. Results for all methods on both rankings	174
7.4. Aspect-specific results of JFSA for predicting the Amazon sales ranking .	175
A.1. List of manually annotated polarity reversing constructions	183
A.2. Detailed results of PRC extraction	184
A.3. Detailed results of word-level consistency classification	185
A.4. Statistical significances for word-level consistency classification	185
A.5. Detailed results of sentence-level polarity classification	186
A.6. Statistical significances for sentence-level polarity classification	187
B.1. Detailed results of the experiments on sentence context	189
B.2. Statistical significances for experiments on sentence context (predicates) .	190
B.3. Statistical significances for experiments on sentence context (arguments)	191
B.4. Detailed results of the experiments on context from task and domain . .	192
B.5. Statistical significances for experiments on context from task and domain	192
B.6. Detailed results of the experiments on annotation design context	193
C.1. Detailed results on development set for LABELED arguments (part 1) . .	195
C.2. Detailed results on development set for LABELED arguments (part 2) . .	196
C.3. Detailed results on development set for argument candidate extraction . .	197
C.4. Detailed results on development set for predicate similarities	198
C.5. Detailed results on test set for system FL-FLAT	199
C.6. Detailed results on test set for system FL-CONTEXT	200
C.7. Detailed results on test set for system FL-CONTEXT-PREDS	201
C.8. Detailed results on test set for system KK-FLAT	202
C.9. Detailed results on test set for system KK-CONTEXT	203
C.10. Detailed results on test set for system KK-CONTEXT-PREDS	204

List of Abbreviations

AMT	Amazon Mechanical Turk
BL	Baseline
CRF	Conditional Random Field
CSRL	Comparison Semantic Role Labeler
GI	General Inquirer
HIT	Human Intelligence Task
IMS	Institut für Maschinelle Sprachverarbeitung
JFSA	Joint Fine-Grained Sentiment Analysis Tool
MPQA	Multi-Perspective Question and Answer (corpus or dictionary)
NB	Naive Bayes
NLP	Natural Language Processing
POS	Part of Speech
PRC	Polarity Reversing Construction
SRL	Semantic Role Labeling
SVM	Support Vector Machines

Acknowledgments

First of all, I would like to thank my adviser Jonas Kuhn, who adopted me despite his many other responsibilities. He was a constant source of new ideas and while only a tiny percentage of them made it into reality, he always gave me something to think about. Without him this document would not have been written. I also would like to thank Hinrich Schütze, who is responsible for bringing me to the IMS and who got me to write the grant application that resulted in this thesis. Thanks also go to Roman Klinger whose ideas resulted in the last chapter of this thesis. Last but not least in the list of advisers, I would like to thank my second reviewer, Manfred Stede, for agreeing to read my work and his helpful comments in the final stages of this thesis.

Besides advisers, I have profited enormously from my colleagues at IMS. The institute employs a lot of great people and I consider myself very lucky to have been among them for a while. Special thanks go to the members of the sentiment group: Christian Scheible, Andrea Glaser, Khalid Al Khatib and Charles Jochim. Additional inspiration for this thesis came from the members of the GCL group, the reading group and the Mensa lunch group. Also, my time would have been boring without the distractions from my office-mates Christian Scheible, Andrea Glaser and Ina Rösiger (I will always remember decluttering for the move to Vaihingen!).

Finally, I could not have done this without the support of my family and friends, especially Laura and my E-Team. I apologize to all of you for talking so much about this thesis. Now it's over! And of course there is Holger, who suffered the most, but always had faith in me.

This research was made possible through support from a Nuance Foundation grant, the DFG (SFB 732, project D7) and the BMBF (CLARIN-D). I am grateful for the work of my annotators: Olga, Kavitha, Cristin, Cornelia, Khalid, Nyamsuren, the participants of the Annotate-athon and the nameless AMT workers.

Abstract

Sentiment analysis deals with methods to automatically analyze opinions in natural language texts, e.g., product reviews. Such reviews contain a large number of fine-grained opinions, but to automatically extract detailed information it is necessary to handle a wide variety of verbalizations of opinions. The goal of this thesis is to develop robust structurally informed models for sentiment analysis which address challenges that arise from structurally complex verbalizations of opinions. In this thesis, we look at two examples for such verbalizations that benefit from including structural information into the analysis: negation and comparisons.

Negation directly influences the polarity of sentiment expressions, e.g., while “*good*” is positive, “*not good*” expresses a negative opinion. We propose a machine learning approach that uses information from dependency parse trees to determine whether a sentiment word is in the scope of a negation expression.

Comparisons like “*X is better than Y*” are the main topic of this thesis. We present a machine learning system for the task of detecting the individual components of comparisons: the anchor or predicate of the comparison, the entities that are compared, which aspect they are compared in, and which entity is preferred. Again, we use structural context from a dependency parse tree to improve the performance of our system. We discuss two ways of addressing the issue of limited availability of training data for our system. First, we create a manually annotated corpus of comparisons in product reviews, the largest such resource available to date. Second, we use the semi-supervised method of structural alignment to expand a small seed set of labeled sentences with similar sentences from a large set of unlabeled sentences.

Finally, we work on the task of producing a ranked list of products that complements the isolated prediction of ratings and supports the user in a process of decision making. We demonstrate how we can use the information from comparisons to rank products and evaluate the result against two conceptually different external gold standard rankings.

Deutsche Zusammenfassung

Sentimentanalyse befasst sich mit Methoden zur automatischen Analyse von Meinungen in Texten wie z.B. Produktbewertungen. Solche bewertenden Texte enthalten detaillierte Meinungsäußerungen. Um diese automatisch analysieren zu können müssen wir mit strukturell komplexen Äußerungen umgehen können. In dieser Arbeit präsentieren wir einen Ansatz für die robuste Analyse von komplexen Meinungsäußerungen mit Hilfe von Informationen aus der Satzstruktur. Wir betrachten zwei Beispiele für komplexe Meinungsäußerungen: Negationen und Vergleiche.

Eine Negation hat direkten Einfluss auf die Polarität einer Meinungsäußerung in einem Satz. Während *“gut”* eine positive Meinung ausdrückt, ist *“nicht gut”* negativ. Wir präsentieren ein System, das auf maschinellem Lernen beruht und Informationen aus dem Satzstrukturbaum verwendet um für ein gegebenes Schlüsselwort festzustellen, ob im Kontext eine Negation vorkommt die die Polarität beeinflusst.

Als zweites Beispiel für komplexe Meinungsäußerungen betrachten wir Vergleiche von Produkten, z.B. *“X ist besser als Y”*. Wir präsentieren ein lernendes System, das die einzelnen Komponenten von Vergleichen identifiziert: Das Prädikat bzw. das Wort, das den Vergleich einführt, die beiden Entitäten, die verglichen werden, der Aspekt in dem sie verglichen werden, und welche Entität als besser bewertet wird. Auch hier verwenden wir Satzstrukturinformationen um die Erkennung zu verbessern. Ein Problem für die Anwendung von maschinellen Lernverfahren ist die eingeschränkte Verfügbarkeit von Trainingsdaten. Wir gehen dieses Problem auf zwei Arten an. Zum einen durch die Annotation eines eigenen Datensatzes von Vergleichen in Kamerabewertungen. Zum anderen indem wir eine halbüberwachte Methode einsetzen um eine kleine Menge von manuell annotierten Sätzen durch ähnliche Sätze aus einer großen Menge unannotierter Sätze zu ergänzen.

Abschließend bearbeiten wir die Aufgabe, den Auswahlprozess eines Kunden zu unterstützen indem wir eine Rangfolge von Produkten erstellen. Wir demonstrieren, wie wir Vergleiche zu diesem Zweck nutzen können und evaluieren unser System gegen zwei konzeptionell unterschiedliche Rangfolgen aus externen Quellen.

1. Introduction

1.1. Motivation

The opinions of others has always been an important factor in decision making for us humans. We watch a movie because of the recommendation of a family member, go to the restaurant our co-worker enjoys, and discuss our life choices with friends. Our objective in each case is to make the choice that is most appropriate for our individual needs, and to this end we hope to take advantage of other people's experience. The emergence of social media has added a new dimension to the exchange of opinions, as countless people express their opinions on review sites, blogs, forums, tweets, social networks and other Web 2.0 pages. While there is obvious benefit in getting opinions on just about everything via the web, at the same time the large number of available opinions for popular products, which often have thousands of reviews, makes it impractical for a human to read them all. Not surprisingly, this development has in recent years generated considerable interest for sentiment analysis (or opinion mining), the area in Natural Language Processing that deals with the automatic analysis of opinions in text.

The most straight-forward task in sentiment analysis is document-level polarity classification (positive, negative), e.g., of product reviews. This task is clear-cut and it is possible to construct new domain-specific data sets for supervised training from review collections, in which the review text almost always goes along with some structured rating, without the need of additional manual annotation. While there is commercial interest in this task and knowing the overall sentiment polarity for a product may sometimes have uses for supporting the choice process of a user, in most actual situations, the relevant sentiment information is to be found on a more fine-grained level.

A large amount of such fine-grained information is contained in the unstructured textual part of a review which usually gives separate judgments on different aspects of an entity, e.g., a camera's weight may be evaluated as negative, while at the same time its picture quality is evaluated as positive. Because not all aspects will be equally important to every user, aspect-based sentiment analysis captures polarity specific to these individual aspects, which may then be aggregated individually. This enables the

users to consider only those aspects that are important for their individual requirements, which greatly enhances the usefulness of sentiment analysis for a decision process.

To reliably gather this type of detailed information, it is necessary to handle a wide variety of complex verbalizations of opinions. The goal of this thesis is to develop robust structurally informed models for sentiment analysis which address challenges that arise from structurally complex verbalizations of opinions. Our underlying hypothesis is that there is no one-size-fits-all approach, but that it is necessary to look at the properties of each of these verbalizations individually and integrate the relevant structural information into the analysis. We select two examples of complex verbalizations to work on in this thesis: negation and comparisons.

Our first example of a complex verbalization is negation, or more generally, polarity reversers. Polarity reversers may be the most intuitively plausible challenge for polarity classification. Many approaches to sentiment analysis are based on sentiment words or phrases which have a prior polarity and serve as indicators for the overall sentiment polarity of a text snippet, e.g., “*good*” expresses positive prior polarity, “*disappointment*” negative polarity. While there are cases where individual words or phrases are sufficient to determine sentiment polarity, sentiment is highly compositional and context may influence and even reverse the polarity of individual sentiment words, e.g., “*not good*” changes the contextual polarity to negative and conversely “*lack of disappointment*” is positive. As already hinted at by these two examples, polarity reversing expressions are diverse and using a fixed list of polarity reversers is not sufficient to cover all of them. Additionally, polarity reversers do not function as reversers in every context and they have a specific scope, e.g., “*not only is it good*” is still positive because “*good*” is not in the scope of the negation. Polarity reversers occur frequently in opinionated text (influencing about 10% of sentiment words in our data), so any fine-grained sentiment analysis system will only produce reliable results if some way of treating polarity reversal is integrated. In this thesis, we propose a structurally informed model to detect whether a given sentiment word in context is in the scope of a polarity reverser.

As the second example of complex verbalizations of opinions, which is the topic of the majority of this thesis, we focus on comparisons like “*X is better than Y*”. For our purposes we define a comparison to be any statement about the similarity or difference of two entities. Besides the linguistic category of comparative sentences, this includes a wide variety of expressions found in user generated texts, such as “*X blows away all others*”, “*X and Y have the same sensor*”, or “*X wins over Y*”. There tends to be a substantial proportion of reviews that include explicit textual comparisons. In our data, about 5%-10% of sentences compare competing products as a whole or in certain aspects.

Also, comparisons are presumably the most useful kind of expression when it comes to supporting a process of choice, as they provide explicit comparative judgments. Still, in standard practice, comparisons are not included among the results of aspect-oriented sentiment analysis systems that generally assign one polarity to one target entity for one sentiment expression. Comparisons do not fit into this scheme and need to be treated differently, as they involve more than one target entity and may assign more than one polarity, e.g., the statement “*X has a better lens than Y*” expresses positive sentiment towards *X* and less positive or maybe even negative sentiment towards *Y*. In this thesis we propose a system that uses structural information to detect the entities involved in the comparison and their relations.

To summarize, in this thesis we address polarity reversers and comparisons as two frequent and important examples for complex verbalizations of opinions. Our aim is to develop methods that use structurally informed models, but at the same time are robust enough to deal with user generated texts, specifically product reviews, which present their own set of challenges due to the use of non-standard language.

1.2. Research questions and contributions

The goal of this thesis is to explore how robust structurally informed models for sentiment analysis can be developed, addressing challenges that arise from structurally complex verbalizations of opinions, specifically polarity reversers and comparisons. Our underlying hypothesis is that there is no one-size-fits-all approach, but that it is necessary to look at the properties of each of these verbalizations individually and integrate the relevant structural information into the analysis. This section outlines the primary research questions that we investigate in this thesis.

We think that both our examples of complex verbalizations would benefit from structural context, but it is unclear which type of context is beneficial and how it can best be integrated into the analysis. We use a syntactic parser, specifically a dependency parser, to integrate structural sentence context. A dependency parser represents syntax as grammatical relations between words, e.g., *A* is subject of *B*. The parser has been trained on news texts that use standardized language. We are working on product reviews which present their own set of challenges due to the use of non-standard language, noisy spelling and punctuation. It is unclear whether applying a standard parser to such user generated texts produces results that are robust enough to be of use for the analysis. If this is not the case, including structural information into the analysis might be hurtful rather than beneficial.

The first and main question we are going to consider in this work is the following:

Research Question A: *How can structural linguistic context information be used for the reliable detection of complex verbalizations of opinions?*

We investigate the topic of including structural linguistic context for both of our examples of complex verbalizations. For polarity reversers, we use a machine learning classifier to distinguish whether a given sentiment word in context is consistent or inconsistent with its prior polarity, i.e., determine whether a polarity reverser is present in the context around the sentiment word. We investigate different ways of representing the relevant sentence context around the sentiment word to see if context from syntactic parsing is more helpful than context from the surface structure alone. We show that for our classifier using dependency path-based features for representing sentence context improves over the window-based context used in previous work (Ikeda et al., 2008).

For comparison detection, we train a machine learning system for the task of detecting the individual components of comparisons: the anchor or predicate of the comparison, the entities that are compared, which aspect they are compared in and which entity is preferred. We experiment with three different types of context: Sentence context, context from knowledge about the task and the domain, and context from the annotation design of the data used for training our system. Similar to our experiments for polarity reversers, we use dependency tree information for sentence context and compare to window-based context used in previous work (Jindal and Liu, 2006b). For comparative predicates, adding both types of context is beneficial, but for arguments structural context clearly outperforms window-based context. To include context from the task and domain, we use generalization techniques to overcome sparsity issues, add information about possible types of comparisons, and include sentiment polarity information. While our experiments on task context yield mixed results, we demonstrate that annotation design decisions, especially those concerning the linguistic anchoring of multiword predicates, have a considerable impact on the overall classification performance and we present a detailed analysis of the effects of the systematic variation of annotations.

Our main contributions towards research question A are as follows:

- We demonstrate that on our data, structural context in the form of paths through the dependency tree is more helpful for a machine learning classifier than window-based context to distinguish whether a given sentiment word in context is consistent or inconsistent with its prior polarity.
- We also find in our experiments that structural context from dependency trees is more helpful than window-based context information for the identification and

classification of comparative arguments.

- We present a detailed analysis of the impact of different contexts due to different decisions taken in the annotation design on the classification performance of our comparison detection system.

Using our structurally informed models, we can make robust predictions for the domain that our system has been trained on, but for different data sets it is advisable to include domain-specific training data due to differences across domains, most notably in the specific vocabulary used for sentiment expressions, e.g., while “*unpredictable*” is positive for the plot of a movie or book, it is negative when used to describe the steering behavior of a car. While investing in the quality-controlled manual annotation of a relatively large amount of training data is the most effective approach to ensure a high-quality system, it may not always be possible to acquire the necessary human and/or financial resources. Since the higher-level structure of comparisons as they appear in reviews is clear-cut, as the syntactic structure of comparisons will be the same across domains and the same semantic constraints apply for the selection of participants, the problem setting could respond favorably to semi-supervised training strategies that start out from a small seed set of manually annotated data which is relatively cheap to create.

The second research question addressed in this thesis is thus:

Research Question B: *How can the structure of complex verbalizations of opinions be exploited in order to automatically annotate training examples by using semi-supervised methods?*

We investigate this question for the task of comparison detection. As our semi-supervised method, we use structural alignment, proposed for Semantic Role Labeling by Fürstenau and Lapata (2009, 2012). This method starts with a small seed set of labeled seed sentences, finds sentences in a large corpus of unlabeled data that are similar to the seed sentences, and projects the labels onto them. By adding the newly annotated sentences to the training data for our system, the initial small manually annotated seed set of sentences is expanded to create a set large enough for efficient machine learning without additional manual annotation effort. Capturing the structure of comparisons is the key step in this process to ensure that the found sentences are similar enough so that the projection of labels onto them produces correctly labeled new training examples, but also that enough linguistic variation is captured to create unseen information for the classifier. We propose adaptations to two steps of structural alignment as a way of tailoring the approach to the structural characteristics of comparisons: the similarity measure and the method of extracting candidates for comparative arguments. Our

results indicate that training on data expanded with these adaptations improves the performance of a system compared to training on data expanded without the adaptations.

Our main contributions towards research question B are as follows:

- We show that we can exploit the structure of comparison to expand a small seed set of labeled comparisons by projecting their structure onto unlabeled sentences using the semi-supervised method of structural alignment.
- We find that the projection can be improved over a generic baseline by using contextual similarity measures and argument candidate extraction methods that take into account the syntactic structure of comparisons.

Finally, recall that our motivation for analyzing complex verbalizations of opinions has been to be able to provide detailed information to support a user in a process of choice, e.g., a purchase decision among a set of candidate products, say cameras A, B and C. One step towards reaching such a decision is to rank the products in question according the product aspects that are relevant for the user’s decision. Textual comparisons provide explicit comparative assessments between products or aspects of products, which makes them presumably the most useful kind of expression for this task, as confounding factors like cultural and personal differences in expressing sentiment are less relevant than in non-comparative assessments. For example, if we have statements that “*A has a better lens than B and C*”, “*B has higher resolution than C*” and “*B is heavier than C*”, we can rank the products according to the user’s preferences as A, B, C if the resolution is more important than the weight or as A, C, B otherwise.

The final research question we want to investigate in this thesis is thus:

Research Question C: *How can information found in textual comparisons be used to support a process of choice?*

We present a method to create a ranking over a set of products by aggregating all individual comparisons found in the data. Specifically, we calculate a score based on the number of times a product is mentioned as the preferred or non-preferred entity in a comparison. We compare these comparison-based rankings to other methods for ranking that are based on subjective phrases and review meta data. We evaluate on two external rankings, a sales ranking, which reflects the number of times an item has been sold in a given time period relative to similar products, and an expert ranking, for which a domain expert compares different products in some specified aspects and ranks them by their quality. While our system performs average when compared to the sales ranking, it achieves the best result for the expert ranking.

Our main contributions towards research question C are as follows:

- We present the task of producing a ranked list of products that complements the isolated prediction of ratings to support an user’s process of choice.
- We show that our method of creating a ranking from textual comparison expressions outperforms all other methods for predicting an expert ranking.

Accompanying this thesis, a dedicated gold standard of manually annotated comparisons has been created and made available to the research community. The corpus is based on the English camera review data by Branavan et al. (2009) and contains 2200 sentences and 2700 comparisons with detailed annotations, which makes it the largest such resource currently available to our knowledge. The data is available from <http://hdl.handle.net/11022/1007-0000-0000-8E72-0>.

1.3. Structure of this thesis

The remainder of this thesis is structured as follows. In **Chapter 2** we provide some background information. The chapter first introduces basic concepts in sentiment analysis and approaches for the automatic detection of sentiment that include structural information. It then describes in more detail the relevant work concerning polarity reversers and comparisons, our two examples of complex verbalizations of opinions.

Chapter 3 discusses our work on polarity reversers relating to research question A. The task we are working on is to distinguish whether a given sentiment word in context is consistent or inconsistent with its prior polarity. We train a machine learning classifier for this task and compare the representation of context as paths through the dependency tree to a window-based approach that uses the local context to the left and right of the sentiment word. Besides evaluating performance directly on sentiment words, we evaluate on the higher-level task of predicting sentence-level sentiment. We also extract the specific reversing paths from large amounts of data, evaluate the results on a set of manually labeled paths and use these paths as features in our machine learning classifier. To enable a more accessible presentation, the chapter contains only a summary of the results, the complete tables for all experiments as well as a list of extracted reversing paths can be found in **Appendix A**.

The remaining chapters 4–7 of this thesis focus on comparisons, presumably the most useful kind of expression when it comes to supporting a process of choice, as they provide explicit comparative assessments of products. Not many manually annotated resources with such detailed sentiment information exist and those have some limitations.

Thus, we create our own manually annotated resource of comparisons in the domain of camera reviews. **Chapter 4** describes the procedure and the annotation scheme used to create the data. We report agreement numbers for our three annotators and compare our data to other corpora in terms of statistics and annotation guidelines.

In **Chapter 5** we address research question A and use our annotated data to train a system for comparison detection which we call CSRL. Our system uses a pipeline of machine learning classifiers and treats the task of detecting comparisons similar to a role-labeling problem. We present a detailed investigation of the effect on the performance of CSRL for different types of context: syntactic context from the containing sentence, context from knowledge about the task and the domain, and annotation design context. We evaluate on our data introduced in Chapter 4, as well as on three other data sets from previous work and discuss some general issues we encounter. **Appendix B** contains the detailed experimental results.

A way around the problem of insufficient training data is to expand a small labeled seed data set with unlabeled data, which is relevant to research question B. In **Chapter 6** we use the semi-supervised method of structural alignment to find similar sentences in a large corpus of unlabeled data. Similarity is compared based on the syntactic and semantic contexts of the predicates. Labels are projected from the seed sentence onto the most similar unlabeled sentences. We first describe the general outline of structural alignment and then introduce several modifications which reflect that our arguments are further away from the predicate and more context is needed to find good expansion candidates. We directly evaluate a part of the found expansion sentences using a labeled development set, and also add the found expansion sentences to the training data for CSRL and compare its performance to training on the non-expanded set. Similar to the previous chapter, detailed experimental results can be found in **Appendix C**.

It is a straight-forward idea to exploit textual comparison expressions to form an aggregated ranking of all products the user is interested in. As the last part of this thesis, **Chapter 7** addresses research question C and presents the task of producing a ranked list of products that complements the isolated prediction of ratings and supports the user in a process of decision making. We demonstrate how we can use the information extracted with CSRL to create a ranking of products by counting the number of times a product is mentioned as the preferred or non-preferred entity in a comparison. To evaluate our method, we discuss appropriate external rankings and present experiments on two different types of rankings, a sales ranking and an expert ranking.

Chapter 8 concludes the thesis, summarizes our findings and contributions and provides an outlook on future work.

1.4. Publications

Parts of the research described in this thesis have been published in:

- Kessler, W. and Schütze, H. (2012). Classification of inconsistent sentiment words using syntactic constructions. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 569–578 (relevant to Chapter 3)
- Kessler, W. and Kuhn, J. (2013). Detection of product comparisons – How far does an out-of-the-box semantic role labeling system take you? In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1892–1897 (relevant to Chapter 5)
- Kessler, W. and Kuhn, J. (2014a). A corpus of comparisons in product reviews. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC '14)*, pages 2242–2248 (relevant to Chapter 4)
- Kessler, W. (2014). Improving the detection of comparison arguments in product reviews. In Plödereder, E., Grunske, L., Schneider, E., and Ull, D., editors, *Lecture Notes in Informatics*, volume 232, pages 2311–2316. Gesellschaft für Informatik, Bonn (relevant to Chapter 5)
- Kessler, W. and Kuhn, J. (2014b). Detecting comparative sentiment expressions – A case study in annotation design decisions. In *Proceedings of 12. Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS '14)*, pages 165–170 (relevant to Chapter 5)
- Kessler, W. and Kuhn, J. (2015). Structural alignment for comparison detection. In *Proceedings of the 10th Conference on Recent Advances in Natural Language Processing (RANLP '15)*, pages 275–281 (relevant to Chapter 6)
- Kessler, W., Klinger, R., and Kuhn, J. (2015). Towards opinion mining from reviews for the prediction of product rankings. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA '15)*, pages 51–57 (relevant to Chapter 7; joint work with Roman Klinger who contributed his system JFSA, the review data and the idea for evaluation)

2. Sentiment Analysis

2.1. Introduction

Sentiment analysis or opinion mining¹ is concerned with the investigation of opinion expressions in natural language. The area has received considerable attention in recent years, which is probably in large part due to the increased availability of opinions in user generated content on review sites, blogs, forums, tweets, social networks and other Web 2.0 pages. Despite the growing problem of fake reviews, users trust reviews of products more than descriptions by the sellers or manufacturers. Also reviews are about actual usage experiences of non-experts that may be similar to the need of the user, so the information is valuable to enable users to make an informed decision about the products they are buying. Sentiment has also been used to predict stock market prices, analyze voter opinions, improve human-computer interaction, and for many other applications, see Pang and Lee (2008) for an overview. The large amount of available opinions for popular products, which often have thousands of reviews, makes reading them all impractical for a human, so automatic systems are desirable.

For whole reviews, document labels can easily be extracted from star ratings, so the creation of large amounts of training data for supervised machine learning is straightforward. The automatic prediction of document polarity (positive, negative) has been the first task of early sentiment analysis systems (Pang et al., 2002). While document level information may be interesting in some cases, in most actual situations, the relevant sentiment information is to be found on a more fine-grained level.

As an illustration, consider Figure 2.1 which shows some of the opinion-related information found on amazon.com for the digital camera Canon EOS 1D X. Reviews on the platform are written by users. The text is accompanied by a star rating, where the user assigns an overall quality rating of between one (very negative) and five stars (very positive). Figure 2.1b shows the average star rating and the distribution of ratings for this specific camera. There are way more positive reviews than negative reviews, which

¹Like most of the sentiment analysis community, we use the terms ‘sentiment’ and ‘opinion’ interchangeably in the following (Liu, 2015).

Electronics > Camera & Photo > Digital Cameras > Interchangeable Lens Cameras > DSLR Cameras

Canon EOS-1D X 18.1MP Full Frame CMOS Digital SLR Camera (Discontinued by Manufacturer)
 by Canon
 ★★★★★ 53 customer reviews
 | 27 answered questions

Price: **\$4,502.42** & **FREE Shipping**

Only 1 left in stock.
Estimated Delivery Date: Tuesday, June 7 when you choose Two-Day Shipping at checkout.
 Ships from and sold by **PORTABLE GUY.**

(a) Canon EOS 1D X basic product information

Customer Reviews



(b) Overview of review star ratings

4 of 5 people found the following review helpful
 ★★★★★ Great camera. The only downside is the shutter noise, August 11, 2014

By [Deandre](#)

Verified Purchase (What's this?)

This review is from: Canon EOS-1D X 18.1MP Full Frame CMOS Digital SLR Camera (Discontinued by Manufacturer) (Camera)

Great camera. The only downside is the shutter noise. Canon really need to fix this issue. My 5d mkIII is not as loud as this monster. Don't get me wrong I love it. I just wish it was a bit on the quieter side.

(c) Example review



THE PROBLEM WITH AVERAGING STAR RATINGS

(d) Comic from xkcd.com

Figure 2.1.: Screenshots from amazon.com of the basic product information for the camera Canon EOS 1D X, the overview of star ratings and an example review for the same product (all taken on June 3rd, 2016); comic from <http://www.xkcd.com/937/> by Randall Munroe.

is typical for product reviews. The average star rating for this camera is 4.1, which may seem high, but at the point of taking the screenshot, 794 cameras in the category “DSLR Cameras” had an average star rating of more than four stars. A total of 185 products even had an exact 5.0 rating. Such high ratings are often based on fewer than five reviews, so the rating is not representative of a group of users.

Besides the unreliability of star ratings, much more interesting information is given in the reviews than can be captured in a simple star rating. The comic in Figure 2.1d makes this point in a humorous way. Based on the star rating alone, a user may think that the “Tornado Guard” is a useful app. Unfortunately, the five star reviews base their evaluation on the user interface (UI) and other relatively minor features. The crucial function of the app, i.e., warning the user about a tornado, is only addressed in the one negative review. In the real world star ratings will usually not be as useless as the comic suggests, but there are real-world examples, such as Figure 2.1c, that point to the same issue. While the review is mostly positive, it also contains a negative statement about the noise the camera’s shutter makes. This information is lost if we only assign a polarity on document level, but may of high importance to a user who plans to buy this camera in order to take pictures in an environment where noise is an issue, e.g., during a concert. Conversely, if the reviewer would have taken that single point as a reason to assign one star to the camera, this evaluation may be irrelevant for a different user who wants to take pictures at a sports event where noise is not an issue.

When we want to extract opinions on a more fine-grained level, we have to deal with a large range of complex verbalizations of opinions. This thesis addresses two frequent types of complex verbalizations of opinions. The first is negation, e.g., “*not as loud*” in the review. Here the negation word “*not*” changes or reverses the polarity of the sentiment expression. To correctly predict the polarity of a sentiment expression, it is crucial to treat negation in some way. Second, sometimes opinions are expressed by comparing the entity under review to a different entity, e.g., “*My 5d mkIII is not as loud as*” in the review. So to fully understand a comparative opinion in a review, besides standard polarity (positive, negative), we need to find out which entities are discussed and how they are evaluated in relation to each other.

The rest of this chapter first gives an overview about basic concepts in sentiment analysis (Section 2.2). The other sections focus on related work. Section 2.3 deals with approaches to automatic sentiment analysis, specifically those that integrate some sort of sentence structure into their processing. We finally discuss in more detail polarity reversers (Section 2.4) and comparisons (Section 2.5), the two examples for complex verbalizations of opinions that this thesis is addressing.

2.2. Basic concepts in sentiment analysis

This section introduces general concepts in sentiment analysis. Here, we introduce the concepts which form the basis for our discussion of complex verbalizations of opinions and related work in the following sections. For more information and more detailed discussions, we refer to Liu (2015) and Pang and Lee (2008).

2.2.1. Subjectivity and sentiment relevance

Statements can be categorized as being subjective or objective depending on their verifiability (cf. Liu, 2015, Chapter 2.4). An *objective statement* expresses some factual information about the world. The truth of the statement can be verified by people other than the one making the statement. In contrast, a *subjective statement* expresses some inner state of mind of an individual, some personal feelings or beliefs. It is not directly verifiable. Consider the following examples:

- (2.1) a. “*This is the best camera I have ever owned.*” (subjective)
b. “*I returned the camera yesterday.*” (objective)

The classification of sentences into these categories is called *subjectivity analysis*. Subjectivity analysis is sometimes used as a step before sentiment analysis which is then only performed on subjective sentences.

Using the categories subjective and objective to distinguish sentences that are interesting and not interesting for sentiment analysis has been criticized, as not every subjective sentence contains an opinion, and objective sentences can implicitly indicate opinions (Scheible and Schütze, 2013):

- (2.2) a. “*I wanted a camera with good voice resolution.*” (subjective)
b. “*The earphone broke in two days.*” (objective)

The first sentence is a subjective wish of the author, but it contains no opinion. The second sentence is objective: we can verify that the earphone did indeed break at the given time. Still, it expresses a negative opinion about the product because the fact that the earphone broke is undesirable. Such sentences are sometimes called *fact-implied opinions* or *polar facts*, and are difficult to distinguish from actual subjective sentiment, even for humans (Ruppenhofer and Rehbein, 2012). In practice, usually all *opinionated sentences*, i.e., all sentences that contain opinions, are selected for sentiment analysis, regardless of whether they are subjective or objective (Liu, 2015). Scheible and Schütze (2013) introduce the notion of *sentiment relevance* for the distinction.

2.2.2. Sentiment polarity and strength

The most salient part of an opinion is the “direction” of the judgment that it expresses, which is called *sentiment polarity* or *semantic orientation*. Sentiment polarity can be positive (indicating praise) or negative (indicating criticism):

- (2.3) a. “*The images on my camera were clear.*” (positive)
b. “*Resolution was not that good.*” (negative)

The value in between the poles is usually assumed to be **neutral** sentiment. Other possible interpretations of this value include **no sentiment** (or objective, see Section 2.2.1) and **mixed** positive and negative sentiment:

- (2.4) a. “*I bought camera XY.*” (no sentiment)
b. “*The camera is mediocre.*” (neutral sentiment)
c. “*The camera is good overall, but has some faults.*” (mixed)

Opinions vary not only in polarity, but also in *strength* or *intensity*:

- (2.5) a. “*I don’t think this is a good camera.*” (weakly negative)
b. “*This camera is a piece of junk.*” (strongly negative)

Sentiment polarity and strength together can be represented by a real number between the two poles of **positive** and **negative** (Polanyi and Zaenen, 2004). Andreevskaia and Bergler (2006) argue that using only one value does not capture all the nuances of polarity. They view sentiment as fuzzy categories, where membership is gradual and some members are more central than others. A word may be a member in several categories. Following the same idea, SentiWordNet (Baccianella et al., 2010) assigns three independent sentiment scores for each word sense: positivity, negativity, objectivity.

Assigning continuous values manually and using them in systems can be challenging, so in practice polarity is usually mapped to discrete categories. The most common classification scheme for sentiment analysis uses only two categories, **positive** and **negative**. Another frequently used category scheme is the star rating of between one (**negative**) and five stars (**positive**) that is used by review platforms. The classification of text snippets into these categories is then called *polarity classification*.

2.2.3. Polarity clues and sentiment dictionaries

Keywords and keyphrases that indicate sentiment are called *polarity clues*. Clues are often adjectives, but can also be nouns (“*rubbish*”), verbs (“*hate*”) or complete phrases (“*cost someone an arm and a leg*”). *Sentiment dictionaries* or *polarity lexicons* list such polarity clues with their associated polarity. The polarity of a word may vary

for different parts-of-speech and different senses of a word. Several manually compiled dictionaries have been presented for English, among them the General Inquirer (Stone et al., 1966) and the MPQA dictionary (Wilson et al., 2005). To avoid the effort necessary for the manual creation of these resources, several methods for their automatic generation have been proposed (Hatzivassiloglou and McKeown, 1997; Turney and Littman, 2003; Andreevskaia and Bergler, 2006).

Polarity clues do not always express sentiment. Conversely it is possible to express sentiment without words that are obvious sentiment words. Consider the following examples from Pang and Lee (2008):

- (2.6) a. “*With great power comes great responsibility.*” (no sentiment)
 b. “*If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut.*” (negative)

Despite two occurrences of the polarity clue “*great*”, no sentiment is expressed in the first sentence. The second sentence expresses a strongly negative opinion about some perfume, even though probably none of its words are contained in a sentiment dictionary.

2.2.4. Prior and contextual polarity

Sentiment is highly compositional and sentence context can influence the polarity of a clue. The polarity of a clue out of context is called the *prior polarity*, the polarity in context is called *contextual polarity* (Wilson et al., 2005). Several contextual factors can influence the polarity of a word. One factor is the domain context, as a word can have different polarities in different domains. The word “*funny*” for example is positive in the movie domain (“*funny plot*”) and negative in the food domain (“*funny taste*”). Another factor is the sentiment target, which is relevant even within the same domain. The word “*long*” in the camera domain for example is positive with some targets (“*long battery life*”), but negative with others (“*takes long time to focus*”). Other factors include cultural differences, personal bias or point of view. For example for a seller “*high price*” may be positive, while for a buyer it is negative.

Finally, sentence context influences polarity and can cause it to shift from one direction to the other or modify the strength. Words that influence the polarity of clues in their context are called *polarity modifiers*, *valence shifters* or *contextual sentiment influencers*. Polanyi and Zaenen (2004) give an overview about many of these modifiers.

A prominent polarity modifier is *negation*, which is also the first complex verbalization of opinion we are going to address in this thesis. Negation can reverse the polarity of an expression from positive to negative or vice versa. We discuss negation and related

work on their treatment in more detail in Section 2.4. A related category are *polarity shifters*, also called *intensifiers/diminishers*, *amplifiers/downtoners*, or *degree modifiers*, which do not influence polarity direction, but weaken or intensify the strength of the sentiment expression. Some examples are “*very*” or “*deeply*” which increase the strength of a sentiment word and “*somewhat*” or “*slightly*” which decrease the strength.

Other sentence-level constructions discussed by Polanyi and Zaenen (2004) that affect polarity direction or strength are *modals*, which express possibilities, permissions or desires instead of actual opinions, *conditional* sentences, which express implications or hypothetical situations instead of actual opinions, or *presuppositional items* which require world knowledge to determine the polarity:

- (2.7) a. “Although you might think the camera is bulky, . . . ”
 b. “If Sony makes good cameras, I will buy one.”
 c. “The battery lasts 2 hours.”

A topic that has gained more attention in recent year is *sarcasm* and *irony*. Tsur et al. (2010) define sarcasm as the activity of saying or writing the opposite of what you mean, or of speaking in a way intended to make someone else feel stupid or show them that you are angry. Sarcasm is quite frequent in reviews, when people try to be funny and/or clever. They give the following examples for sarcastic sentences:

- (2.8) a. “As much use as a trapdoor on a lifeboat.”
 b. “This book was really good – until page 2!”

2.2.5. Sentiment targets and aspect-oriented sentiment analysis

Sentiment does not only have a polarity, but it is also expressed with respect to some particular entity or object usually called the *sentiment target* (Hu and Liu, 2004). Target detection is important because one text may contain opinions on several different products. Targets are assigned for individual sentiment expressions because several products or several aspects of a product may be discussed in a single sentence. Sentiment analysis approaches that identify a target along with a sentiment expression are called *fine-grained*, *target-specific* or *aspect-oriented*. Consider the following part of a review with assigned sentiment polarities and targets (Liu, 2015):

- (2.9) a. “I bought a Canon G12 camera six months ago.” (no sentiment)
 b. “I simply love [it]_E.” (positive, Canon G12)
 c. “The [picture quality]_A is amazing.” (positive, Canon G12 picture quality)
 d. “The [battery life]_A is also long.” (positive, Canon G12 battery life)
 e. “However, my wife thinks it is too [heavy]_A for her.” (negative, Canon G12 weight)

The expressed sentiment is positive for the Sentences 2.9b–2.9d and negative for the last sentence. In the second sentence, we have a positive sentiment expressed towards the camera “*Canon G12*” as a whole, the *target entity* (also called *topic*). The following sentences express sentiment about an *aspect*² of the same camera: the “*picture quality*”, “*battery life*” and “*weight*”. The target of a sentiment expression can thus be formalized as a pair of entity and aspect, both of which may be empty or implied by the context. The aspect can also be implied by the sentiment expression (*implicit aspect*), e.g., in Sentence 2.9e, the sentiment word “*heavy*” implicitly refers to the aspect “*weight*”.

The definition of what is considered an aspect is usually made on a pragmatic basis. Aspects can be anything related to the entity we are interested in. For products, this usually includes parts or components (e.g., lens, flash, battery of a camera), properties or attributes of the product (e.g., size, weight, price), actions performed with the product (e.g., taking pictures in low light), and results of using it (e.g., the pictures).

Aspects can be used for aspect-based summaries of opinions, e.g., to show that while a camera’s resolution is evaluated as positive, the zoom is evaluated as negative. Most of the time, the set of all extracted aspect phrases will be too fine-grained to be useful and some grouping needs to be done. One reason to group aspect phrases together is that one aspect can be referred to by different phrases, e.g., “*image quality*”, “*photo quality*” and “*picture quality*” all refer to the same aspect. In the easy cases this is an issue of identifying synonyms (e.g., “*image*” and “*picture*”) or orthographic variations and abbreviations (e.g., “*picture*” and “*pics*”). The second issue for grouping aspects is the fact that aspects form a hierarchy, e.g., “*battery life*” and “*battery charge time*” are different aspects of *battery* which in turn is an aspect of a camera. In a summary, these sub-aspects may be grouped together or not, depending on the interest of the user.

A few additional points can be made with regard to fine-grained sentiment analysis. Distinguishing the *opinion holder* may be important for some applications, as not all of the opinions that are expressed may be from the author of the review, e.g., “*my wife liked the camera*”. Similarly, the *time* when the opinion was expressed may be relevant as opinions and circumstances may change over time, e.g., a negative opinions about the price of a camera may not be relevant years later, when the price has become considerably lower. And finally, as companies, organizations and individuals have recognized the importance of opinions particularly for selling products, *opinion spam* has become an issue. These points will not be further investigated in this thesis. For an overview about the issues, please refer to (Liu, 2015).

²Early work in sentiment analysis used the term *feature* instead of *aspect*, but this may lead to confusion with the features used for machine learning, so *aspect* is the preferred terminology.

2.3. Structurally informed sentiment analysis

This section presents some related work in the area of sentiment analysis that deals with the integration of structural information into the processing. Automatic sentiment analysis covers a broad range of tasks and approaches, including sentiment polarity classification, target identification, opinion holder identification, sentiment summarization, and emotion classification. Many different methods have been applied and, the integration of context information has been approached in many different ways.

From the various possible types of context (e.g., sentence structure, discourse information, domain knowledge), we put our focus in this section on structural linguistic context in the form of parse tree information, as this is the type of structural context we address in this thesis. This section is not an exhaustive survey of works that use parse tree information. We concentrate mainly on polarity classification and mention some of the more prominent approaches where sentence structure plays an important role. For more information, please refer to the survey by Liu (2015).

2.3.1. Structure in rule-based approaches

Rule-based approaches to sentiment analysis are usually based on polarity clues from a sentiment dictionary (see Section 2.2.3). The sentiment of a text snippet is calculated by counting the polarity clues with their polarity, if there are more positive words, the overall sentiment is **positive**, if there are more negative words, the sentiment is **negative**. This *term counting* or *lexicon-based* approach was first introduced by Pang et al. (2002) and has been widely used as a baseline in sentiment analysis since. Approaches that include structural information either use this information to calculate contextual polarity for every found sentiment expression or to calculate sentence polarity by propagating the polarity from the leaves to the root through a parse tree.

Moilanen and Pulman (2007) present a propagation approach to calculate sentence polarity with the help of manually defined compositional patterns on top of a constituency parse. They use a sentiment dictionary where each entry has one of the tags: non-polar/neutral, negative, positive and polarity reversers. Patterns are used to propagate sentiment through the parse tree, so that each phrase is assigned a sentiment polarity. The patterns are dependent on POS tags and include rules for polarity reversers.

Shaikh et al. (2007) present detailed rules to determine sentiment on sentence level that work on top of a what they call “semantic parse”, i.e., the output of a dependency parser that yields triplets of subject-verb-object according to each semantic verb frame of

the input sentence. Calculation rules take into account POS tags, polarity information, negation and intensifiers/diminishers.

One approach to calculate contextual polarity for individual sentiment expressions is presented by Choi and Cardie (2008). They use hand-written rules motivated by compositional semantics that work on syntactic patterns on top of a constituency parse, e.g., the pattern “VP NP” for “*destroyed the terrorism*”. A rule specifies how the parts of the pattern are to be combined in terms of their polarities, e.g., for the above example as “*destroyed*” is a negator, it would flip the negative polarity of “*terrorism*” to result in positive sentiment. The proposed rules deal with negation and conflicting polarities.

Similarly, Liu and Seneff (2009) present an approach that uses what they call a “parse-and-paraphrase” paradigm. All sentences are parsed by a lexicalized context free grammar which results in what they call “frames that encode semantic dependencies”. These parses are used to extract adjective-noun pairs which are then assigned a polarity and used for the final task, sentiment polarity classification of phrases.

An approach that is not purely rule-based, but makes use of manually defined patterns is proposed by Kanayama et al. (2004). They view sentiment analysis as translation from text to “sentiment units” consisting of polarity and target. They use an existing machine translation system that works on complete dependency parses and performs top-down pattern matches on the tree structures. They replace the translation patterns with sentiment patterns and the bilingual lexicon with a sentiment polarity lexicon.

2.3.2. Structural features for machine learning

Machine learning for polarity classification can be approached the same way as other text classification tasks. Commonly used features include unigrams, bigrams, as well as polarity clues. In this subsection we look at how parse tree information has been included into feature sets. Approaches that are directly based on sentence structure are discussed in the following subsection.

Parse-tree information can be broken down into the individual relation between two words in a sentence. As one of the first to address the task of polarity classification with machine learning, Dave et al. (2003) attempt to determine the polarity of product reviews. They experiment with a wide range of features: WordNet, negation, POS tags, and parse features based on a dependency parser. Their parse features are triples of a word, the dependency relation and the head word, e.g., “*nice(A):subj:camera(N)*”. They report that including these features hurts their baseline of using unigrams only.

Gamon (2004) uses linguistic features for polarity classification of short customer

feedback items. His features include POS trigrams, constituency parse tree information in the form of patterns, grammatical roles, and “logical form” features such as transitivity of a predicate or tense information. In contrast to Dave et al. (2003), his results indicate that linguistic features are beneficial for classification.

Ng et al. (2006) incorporate dependency information into a polarity classifier for movie reviews. In contrast to Dave et al. (2003), they include only specific relations, namely all subject-verb, verb-object and adjective-noun tuples found in the parse tree, e.g., “(*like, movie*)”. Using these dependency-based features improves over unigrams, but not over higher-level n -grams (bigrams and trigrams). In their investigation they hypothesize that their stemming of the words may be problematic, as it does not allow to distinguish between “*he likes the movie*” and “*I like the movie*”.

Based on the mixed results from previous work, Joshi and Penstein-Rosé (2009) investigate the role of dependency relations as features in more detail, although for the task of identification of opinionated sentences. They convert dependency relations into triples of relation-head-dependent and experiment with replacing the lemma of either the head, the dependent or both by the POS tag, e.g., “*amod-NN-great*”. They report the best results by replacing the head word, which they argue yields more generalizable patterns as most of the time the sentiment words are modifiers and targets are heads.

As an alternative to general parse tree relations, features tailored to the specific task can be used. One prominent example is presented by Wilson et al. (2005) who do subjectivity and polarity classification of individual sentiment expressions from news data. Wilson et al. (2009) investigate the effect of the different feature groups in more detail. They find negation to be the most important feature for polarity classification, but this feature is based on window context in their implementation. Next in importance are dependency-based features that model whether a subjectivity clue modifies the current word or the current word is modifying a subjectivity clue. Less important are their other dependency features which indicate whether a subject, copula or passive relation is found on the path from the current word to the root.

Besides individual relations, complete paths through the tree can be used as features which is especially important if polarity classification is addressed in combination with the identification of target or opinion holder. J. Kessler and Nicolov (2009) work on the task of linking already identified sentiment expressions and their targets. They present an approach with features based on syntactic paths through the dependency tree which outperforms manually defined patterns.

More recently, Sayeed et al. (2012) extract triples of sentiment expression, target, and holder in a probabilistic framework. Their features are based on dependency parses and

include POS tags, dependency relations and features of parent and child nodes. They conclude that using more linguistic features increases the stability of the results.

An extensive analysis of different features used for the identification of polarity, holder and target is presented by Johansson and Moschitti (2013). Besides the path between the expressions in the dependency tree, they use the output of semantic role labeling and add features for predicate and argument labels. Their detailed analysis shows that features derived from grammatical and semantic role structure can be used to improve all three sentiment tasks that they are working on.

2.3.3. Machine learning guided by syntax trees

Feature extraction on syntactic structures is difficult and not all information can be modeled this way. So instead of adding features based on parse-tree information, syntax information can be used to guide a machine learning approach, similar to the way rule-based systems propagate sentiment information through the parse tree.

One way of including complex syntax information is the use of tree kernels which efficiently compare tree structures without explicitly extracting features. Tu et al. (2012) work on the task of document-level polarity classification of movie reviews with tree kernels. They focus on selecting the most relevant substructures to keep the feature space small and propose to include only structures around polarity clues with their direct head and the dependents. They investigate different types of kernels and report the best results with dependency trees, lexical information and their filter.

An example for a more direct modeling of syntactic structure is Nakagawa et al. (2010) who use conditional random fields and dependency trees for polarity classification on sentence-level. They introduce hidden random variables that represent the polarity for every node in the dependency tree. The random variables are connected if the corresponding nodes in the parse tree have a dependency relation, so the polarity is propagated through the tree. Their features include the polarities of the subtrees, prior polarities from a sentiment dictionary, polarity reversal and POS tags. They report significant improvements over a term counting baseline.

An approach that has received much attention recently is the work of Socher et al. (2013) who present recursive neural tensor networks for sentence polarity classification. In this deep-learning framework, words are represented as real-valued vectors of a fixed dimensionality. The polarity of phrases is calculated based on the structure given by a constituency tree. Their work also presents the Stanford Sentiment Treebank, a parsed version of the movie review data set (Pang and Lee, 2005) where every phrase of the con-

stituency trees has been labeled with sentiment polarity through crowdsourcing. They report very good classification results on this data and also show with some examples that their model is able to capture linguistic phenomena such as negation.

2.4. Negation and polarity reversers

Negation (negators, negatives, polarity shifters, polarity reversers, ...) is arguably the most intuitively understandable challenge for determining the sentiment polarity of a text snippet. While the word “*good*” expresses positive sentiment, most people will agree that “*not good*” expresses negative (or at least more negative) sentiment. Negation occurs frequently and is a major source of errors in polarity classification. As a result, it has received attention in the sentiment community since the early days of the field (Das and Chen, 2001; Pang et al., 2002; Polanyi and Zaenen, 2004).

For sentiment analysis systems, negation treatment can be split into three steps or tasks (Li et al., 2010; Benamara et al., 2012):

1. negation identification, the task of finding negation expressions,
2. negation scope detection, the task of identifying the words affected by the negation,
3. negation processing, the task of applying the negation effect to its scope.

Consider the sentence “*This camera is not very good, complete rubbish*” as an example. Step one identifies the negation “*not*”, step two determines the words “*very good*” as its scope, and step three reverses the polarity of the sentiment expression from **positive** to **negative**. We discuss approaches for each of these tasks in the following. To our knowledge, there is no sentiment analysis system that addresses all three tasks in a systematic way, most use heuristics for at least one step.

The topic of negation is not only relevant to sentiment analysis, it has been widely studied in linguistics, logic and philosophy (Pullum and Huddleston, 2002). Computational approaches have been researched in various field, especially information extraction for diverse application domains such as medicine (Rokach et al., 2008), biomedical texts (Morante et al., 2008), Wikipedia (Farkas et al., 2010), or soccer (Farkas, 2011). We concentrate on related work from sentiment analysis in the following.

2.4.1. Negation identification

The first task for negation treatment is to identify whether or not any negation is present. While in English, negation is most frequently expressed by using “*not*”, there is a variety of words from all parts of speech that affect polarity the same way (Wiegand et al.,

2010). Some examples include (polarity reversers are underlined): “no problem”, “lack of quality”, “without worrying”, “failed to impress”, “never worked”.

In English, negation is marked by affixes or syntactic markers. To express negation at the morphological level, words are modified by prefixes (e.g., “*un-*”, “*dis-*”, “*in-*”) or suffixes (e.g., “*-less*”). Morphological negation is productive and a relevant topic for constructing sentiment dictionaries, but has otherwise mostly been ignored in sentiment analysis. Negation can be intersentential, but sentiment analysis has mostly focused on negations within a single sentence, which are more frequent (Councill et al., 2010).

Linguistics defines several types of negations: analytic vs. syntactic, verbal vs. non-verbal, clausal vs. subclausal (Pullum and Huddleston, 2002). Sentiment analysis research has ignored these categorizations. In a few cases it introduced some of its own categories. Choi and Cardie (2008) distinguish between function word negators (e.g., “*not*”, “*never*”) and content word negators (words that have negation as part of their semantics, e.g., “*fail*”). In their experiments using only function-word negators is worse than no negation treatment at all. Only by including content-word negators can the baseline be beaten. Wilson et al. (2005) distinguish between general polarity shifters that act on sentiment words of both polarities, negative polarity shifters that make any polarity negative (“*lack of X*”) and positive polarity shifters, that make any polarity positive (“*abate*”). They do not evaluate the influence of this distinction on the results.

For systems that address the task of identifying negations, the most common approach is to use a fixed list of negation words or phrases (Pang et al., 2002; Councill et al., 2010; Taboada et al., 2011; Zhu et al., 2014). Most of the lists are not very long, e.g., Zhu et al. (2014) use a list with only four entries: “*not*”, “*no*”, “*never*”, “*n’t*”.

Using a fixed list of phrases has several limitations (Wiegand et al., 2010). The first limitation is coverage, as negation expressions are diverse. The second is ambiguity, as some expressions do not function as a negation in every context. In the following example from Wiegand et al. (2010), the phrase “*not only*” does not reverse the polarity of the negative sentiment words, despite the presence of the negation “*not*”, the overall sentiment of the sentence is still negative:

(2.10) “[*Not only*] is this phone *expensive*_{neg} but it is also *heavy*_{neg} and *difficult*_{neg} to use.”

The alternative to manually created lists is the automatic extraction of negation expressions, but work in this area is limited. The first somewhat automatic extraction, albeit from manually created resources, is the work of Choi and Cardie (2008) who use words from the categories “NotLW” and “Decreas” of the General Inquirer dictionary (Stone et al., 1966) and add synonyms of the extracted words from WordNet.

A corpus-based approach is presented by Liu and Seneff (2009) who automatically assign a shifting value to every adverb in their data. This shifting value is calculated as the difference between the polarity scores of an adjective alone and the combination of this adjective with the adverb. Polarity scores in turn are calculated as the average rating of all reviews that contain the adjective or the combination. Examples of adverbs that are assigned strongly negative shifting values are “*not*”, “*a little*”, and “*a bit*”. They use the extracted shifting values to calculate polarity in their system, but do not independently evaluate this part of their work. Only adverbs are considered as shifters.

A conceptually different approach to the task of negation identification is presented by Ikeda et al. (2008). Instead of identifying negations and then deciding which sentiment words are in the scope of each detected negation, they use sentiment words as their starting point and determine for each sentiment word whether a negation occurs in its context. The context they use includes the three words to the left and right of the target sentiment word. They evaluate on sentence-level polarity classification and present two ways to include the information of reversing contexts into the classification: either word-wise for each sentiment word, or with one feature vector for each sentence that is the sum of all word-wise information. Li et al. (2010) extend that method to document-level polarity classification by stacking two classifiers that are trained on automatically extracted reversed and non-reversed sentences respectively.

2.4.2. Negation scope detection

After a negation has been found, the next step for a sentiment analysis system is to determine its scope. Scope refers to the part of the meaning that is affected by the negation (Pullum and Huddleston, 2002). Consider the following example sentence.

(2.11) “*It is not that **big**_{neg}, it **comfortably**_{pos} fits in my pocket.*”

Even though “*not*” acts as a reverser for “*big*”, it does not reverse the positive polarity of “*comfortably*” which is outside the scope of the negation. Besides scope, negation expressions also have a focus, the element of the scope that is most prominently or explicitly negated. Focus is even more difficult to detect than scope, and is usually ignored in sentiment analysis. Scope and focus are primarily semantic, but syntax helps in detecting them (Blanco and Moldovan, 2011).

Most approaches in sentiment analysis that do some negation treatment use rather simple heuristics to detect scope. A popular approach assumes the scope to be from the negation expression until the next punctuation mark (Pang et al., 2002). Variations on this approach use the complete sentence (Hogenboom et al., 2011) or a window of fixed

token size around the negation (Hu and Liu, 2004). Pröllochs et al. (2015) use POS tags to further limit the scope to include only matching words from inside one of the above heuristic scopes, e.g., only adjectives are negated.

Some scope detection approaches include syntax information, e.g., Liu and Seneff (2009) use the remainder of the containing clause as scope. Other rule-based approaches use manually defined compositional patterns on top of a dependency parse (Shaikh et al., 2007) or constituency parse (Kennedy and Inkpen, 2006; Moilanen and Pulman, 2007; Choi and Cardie, 2008). Jia et al. (2009) present a system of detailed rules based on dependency parse tree information and linguistic knowledge. Their scope always starts at the negation expression and the rules are used to find a right delimiter, e.g., scope ranges until the next sentiment noun in the direct object of the negated verb. A simplified version of their rules is to take the next sentiment word as the delimiter. Results for all of these heuristics are mixed, but generally more involved methods of determining scope do not yield a significant improvement over using a window of fixed size.

The first machine learning approach for scope detection in sentiment analysis is Council et al. (2010). Their system uses dependency parses and a first-order linear-chain conditional random field to learn the scope of negation in reviews and biomedical texts. Their features include tokens, POS, distance from the closest negation in both token distance and in the dependency tree, and information from the dependency head. They report significant improvement on sentence-level polarity classification compared to using no negation detection at all, but do not compare to other approaches for scope detection. In similar experiments on financial news by Pröllochs et al. (2015), a Hidden-Markov-Model prediction system does not perform well at all compared to different rule-based techniques including next-sentiment-word and window-based scopes.

Outside of sentiment analysis, the task of negation scope detection has received attention in information extraction, especially for the biomedical community as evidenced by shared tasks on the topic, e.g., “Learning to detect hedges and their scope in natural language text” at CoNLL 2010 (Farkas et al., 2010), or “Resolving the scope and focus of negation” at *SEM 2012 (Morante and Blanco, 2012).

2.4.3. Negation processing

The final step for the treatment of negation expressions after they have been identified and their scope has been determined, is negation processing or applying whatever effect negation has to the words in the scope of the negation. In the following, we discuss typical approaches for negation processing first for rule-based and second for machine learning-

based systems. We use the following example sentences and assume that the scope of the negation “*not*” is the word “*like*” in Sentence 2.12a and “*horrible*” in Sentence 2.12b:

- (2.12) a. “I do not *like*_{pos} this new Nokia model.”
 b. “it’s not that *horrible*_{neg}.”

Rule-based systems generally use polarity clues and their polarity scores from a sentiment dictionary as a starting point. To determine the polarity of a sentence, the number of positive and negative clues is counted and the majority class assigned. Applying a negation corresponds to reversing or shifting the polarity scores of the clues in the scope of the negation. For the sake of example, assume polarity values for sentiment words on a scale of $[-3, +3]$ and that the sentiment score of “*like*” is a fairly positive $+2$, the score of “*horrible*” a very negative -3 . As there is only one polarity clue in each example sentence, the overall sentence polarity values without considering the negation would be $+2$ (fairly positive) and -3 (very negative) respectively.

Most commonly, negation is treated as causing polarity reversal (*flip negation*), i.e., a negation changes the polarity from positive to negative or vice versa (Polanyi and Zaenen, 2004; Choi and Cardie, 2008). For continuous values of polarity, flip negation amounts to changing the sign from $-$ to $+$ or vice versa. In the example, this will give a fairly negative polarity of $2 \cdot (-1) = -2$ for Sentence 2.12a and a very positive polarity of $(-3) \cdot (-1) = +3$ for Sentence 2.12b.

While for two classes flip negation works nicely, flip negation in combination with polarity strength leads to undesired effects. The negated expression has the same strength as the non-negated expression, e.g., “*not horrible*” is strongly positive, which arguably does not fit the interpretation of a human reader. To rectify this situation, later works (Liu and Seneff, 2009; Taboada et al., 2011) propose a *shift negation*, i.e., a negation shifts the value to the opposite polarity by a certain factor. While the polarity will still be reversed (at least in all but the most extreme cases), the resulting strength of the negated expression is closer to a neutral value. In the example, let us assume a shifter value of -3 for “*not*”. Sentence 2.12a would receive a slightly negative polarity of $2 + (-3) = -1$. While the resulting sentence polarity is still negative just like it was with flip negation, it is much less strong. Sentence 2.12b receives a polarity of $(-3) - (-3) = 0$, i.e., neutral, which arguably fits a human understanding of “*not horrible*” better than the very positive polarity that results from applying a flip negation.

Shift negation allows for different negation expressions to have different shifter values. While Taboada et al. (2011) use the same shifting value for all negation expressions, Liu and Seneff (2009) calculate an individual value for each of the adverbs they consider. Neither work evaluates the effect the different shifting values have on the results.

Taboada et al. (2011) compare to using flip negation, but find no significant difference. They also perform an empirical study where humans are asked to evaluate whether a negated positive word is more or less strongly negative than a negative word. The results are not conclusive to decide between shift and flip negation. More recently, Zhu et al. (2014) present an empirical study in which they analyze differences in manually annotated polarity values between a phrase and that same phrase combined with a negation expression. They find significant differences in the shifting effect of negators, some of these effects are also depending on the argument polarity.

Negations do not only occur in isolation, multiple negations can be used together. The usual approach for treating multiple negations is to treat each one separately. So if negation reverses the polarity of a statement, two negations would cancel each other out. If negation shifts the polarity score, two negations shift twice, in opposite directions. Choi and Cardie (2008) use the following examples as a justification of this approach:

- (2.13) a. “*I **doubt**_{neg} it.*” (negative)
 b. “*I did not have any **doubt**_{neg} about it.*” (positive)
 c. “*The report eliminated my **doubt**_{neg}.*” (positive)
 d. “*The report could not eliminate my **doubt**_{neg}.*” (negative)

While this may cover the majority of cases, multiple negations are not always independent of each other. Consider these examples from Blanco and Moldovan (2011):

- (2.14) a. “*She is not **unhappy**_{neg}.*”
 b. “*She hasn't eaten nothing.*”

The first example does not state that the person is happy, but that she is somewhere in between the negated and non-negated state, between happy and unhappy, not fully unhappy but not fully happy either (Blutner, 2004). In Sentence 2.14b, while there are two negations, the sentence is interpreted as being negated only once (*negative concord*). The effect in this case is an intensification of the negation. Negative concord is widespread across many English varieties.

Automatically determining which effect a given negation in a given context has is difficult and to our knowledge no system that includes this type of processing has been proposed yet. What has been done is to use a binary model of negation on a global level, i.e., polarity is reversed once, no matter how many negations are found. Choi and Cardie (2008) report no significant difference between the two approaches in their experiments. Conversely, Pröllochs et al. (2015) do report significant improvements when treating each negation individually as compared to a binary negation.

For systems that are based on machine learning, the most common way to incorporate polarity modifiers into a bag of words representation is to represent a token x in the

scope of a negation as the feature NOT_x (Das and Chen, 2001; Pang et al., 2002). If in Sentence 2.12a we assume the scope of the negation to be “like”, the resulting bag of features for the sentence is $\{I, do, NOT_like, this, new, Nokia, model\}$. With this modification, a negated occurrence of a sentiment word does not confuse the classifier, as they are two different features. The disadvantage with this approach is that there is no connection for the classifier between the original word and the modified feature, even though they represent the same word. The method has been used only in combination with heuristic scope detection, e.g., until the end of the sentence, resulting in $\{I, do, NOT_like, NOT_this, NOT_new, NOT_Nokia, NOT_model\}$. The wide scope introduces a lot of noise, so the classification improvement that can be obtained with this method has been reported as being limited (Wiegand et al., 2010). Somewhat similarly, Xia et al. (2016) replace a negated word by a different word that has the same absolute feature weight in the classifier, but for the opposite class. In their experiments, this approach gives approximately the same results as the usual processing.

An implicit way of modeling negation that has proven to be effective in sentiment analysis is via higher order word n -grams, e.g., bigrams (“not like”) or trigrams (“not that horrid”). Such approaches can achieve good performance, not only because of the included modeling of negation, but also because n -grams implicitly cover other polarity-influencing phenomena (Pang et al., 2002; Wang and Manning, 2012).

More explicit negation modeling for machine learning has been proposed in the form of special features to indicate the existence of negations in the context of a polarity clue. Wilson et al. (2005, 2009) report significant classification improvement when they include features to model negation in a window around the phrase to be classified. Choi and Cardie (2008) report improved results when they include the result of term counting with and without including negation in addition to features that include the presence of negators. Nakagawa et al. (2010) integrate polarity reversers into a dependency tree-based method for polarity classification by using features that indicate for each dependent whether it is a reverser, but do not evaluate the influence of this specific feature on their results.

2.4.4. Related phenomena

Some other phenomena have influences on polarity that are similar to negation. Consider the following examples adapted from (Wiegand et al., 2010):

- (2.15) a. “I find the functionality of the new phone [less] **practical**_{pos}.”
 b. “Perhaps it is a **great**_{pos} phone, [but] I thought it was **horrible**_{neg}.”

c. “It [should] have **worked**_{pos} even under water.”

Sentence 2.15a contains a diminisher (e.g., “less”, “hardly”, “slightly”, also sometimes called polarity shifters, intensifiers, approximate negators or downtoners). Diminishers shift the sentiment polarity of the affected polarity clue towards the opposite polarity like negation does, but the influence is usually considered to be not as strong. In sentiment analysis systems, diminishers are most often ignored entirely. If they are considered, they are treated as decreasing the polarity value of a following sentiment word by a small amount (Polanyi and Zaenen, 2004; Taboada et al., 2011), or the same way as negation (Kennedy and Inkpen, 2006; Liu and Seneff, 2009). The distinction between diminishers and negations is not clear-cut. In their empirical analysis of negations and their effect on sentiment, Zhu et al. (2014) find three diminishers that have an effect of the same magnitude as negations: “barely”, “unlikely”, “superficial”.

Sentence 2.15b is an example of a sentence contrast or discourse connectors (e.g., “but”, “although”). These have not been researched in much detail, but some very different approaches of incorporating them into the analysis have been proposed. The earliest work, Polanyi and Zaenen (2004), considers a contrasting discourse connector to basically eliminate the sentiment contribution of one clause, so in the example in Sentence 2.15b the word “great” would be ignored and only the second clause is used to calculate sentence polarity. Taboada et al. (2011) treat “but” like an intensifier for the following clause, so in the example the score of “horrible” would be multiplied by an intensification factor. This treatment is empirically validated by Socher et al. (2013), who show that their neural network learns that the polarity of the clause following “but” dominates the polarity of the complete sentence. Zirn et al. (2011) treat contrasting discourse connectors as reversing the polarity of the preceding clause, in the example that would mean that both sentence parts count as negative.

Modal verbs (e.g., “should”, “might”, “would”) like in Sentence 2.15c express possibilities or set up a context where opinions are expressed that do not reflect the actual opinion of the author. Similarly, conditional sentences (e.g., “if”, “unless”) describe implications or hypothetical situations and their consequences. As a result, the most common treatment is to set the sentiment value to neutral if a sentiment word occurs in the same clause as a modal verb, a conditional marker and other irrealis markers (Polanyi and Zaenen, 2004; Taboada et al., 2011). Narayanan et al. (2009) validate this approach with their analysis of conditional sentences where they find that conditional clauses contain few opinions. Benamara et al. (2012) report in their analysis that modals influence the strength of opinions. Modal verbs can also be used as indicators to detect customer wishes (Goldberg et al., 2009; Ramanand et al., 2010).

2.5. Comparisons in product reviews

Comparisons are the second example of complex verbalizations of opinions that we focus on. Comparisons are the topic of the majority of this thesis. Consider the following examples of opinion expressions about the camera “D200”:

- (2.16) a. *“I was impressed by the fast shutter speed of [D200]_E.”*
 b. *“[The D200]_E can shoot much faster than [my old camera]_E . . .”*

Sentence 2.16a expresses direct positive sentiment about the camera’s shutter speed, so generally an aspect-oriented sentiment analysis systems will assign the polarity **positive** to the target entity “D200” or, more precisely, to the aspect “shutter speed” of that entity. Sentence 2.16b is also a statement about that same camera’s shutter speed, but there are two entities involved in the sentence. Generally, aspect-oriented sentiment analysis systems assign one polarity to one target entity, so what they may do is to ignore the information about the second entity and just give the same result as for the first sentence. This misses important information about the second entity, so one might think to remedy this situation by assigning negative polarity to this entity. But it is questionable whether a comparison expresses this type of sentiment towards the two entities. The fact that X is better than Y does not entail that X is good, nor that Y is bad (Huddleston, 2002; Liu, 2015). As an illustration, consider two possible continuations of Sentence 2.16b:

- (2.17) a. *“... but still not fast enough to capture my daughter on her skateboard.”*
 b. *“... but really both are just amazing!”*

To assign positive polarity to “D200” would be incorrect in Sentence 2.17a, just as it would be incorrect to assign negative polarity to the second entity “my old camera” in the Sentence 2.17b. So instead of assigning sentiment to the two entities, we need to extract the relation of the two entities to each other. Thus, comparisons cannot be treated the same way as other sentiment expressions.

Comparisons are relatively frequent in product reviews (5%-10% of sentences in our data). Some comparisons are among competing products as a whole, most compare a certain aspect of the two products. Comparisons are of interest for companies that do not only want to know what aspects of their product users like or dislike, but also where they stand in relation to their competitors. Also, comparisons are presumably the most useful kind of expression when it comes to supporting a process of choice, as they provide explicit comparative judgments which may enable the user to make a decision between several products. Comparisons are also less influenced by confounding factors like cultural and personal differences in expressing sentiment than other sentiment

expressions. In some cultures “*not bad*” may be the highest praise, and in some other culture the expression “*awesome*” may indicate rather average performance. But when two entities are compared, an explicit ranking is given that cannot be misinterpreted.

The treatment of comparisons can be split into at least three tasks:

1. comparison sentence identification or the task of detecting if a given sentence contains a comparison,
2. comparison component detection or the task of identifying the entities and other parts that are involved in the comparison, and
3. comparison type identification or the task of deciding what type of comparison it is and what ordering of entities is introduced.

Unlike negation, only few approaches treat comparisons in a sentiment analysis context. We describe these approaches in the following subsections, structured by the task or tasks that they are designed to perform.

2.5.1. Comparison sentence identification

Comparisons and their syntax and semantics have been widely studied in linguistics (Heine, 1997; Huddleston, 2002; Cuzzolin and Lehmann, 2004; Kennedy, 2010; Stassen, 2013). Comparisons are used to position two entities relative to each other, i.e., to express their similarity or dissimilarity. Like many other languages, English grammar has specialized morphology and syntax for such expressions, most prominently the comparative and superlative form of adjectives and adverbs, e.g., “*big*”, “*bigger*”, “*biggest*”. Classic examples of comparative sentences as investigated by linguists are those that contain such a word form and some other constructions used specifically to express comparisons:

- (2.18)
- a. “*X is bigger than Y*”
 - b. “*X is the biggest*”
 - c. “*X is as good as Y*”

There are a few constructions that look similar and contain comparative keywords, but do not involve a comparison of two objects. Consider these examples from (Huddleston, 2002; Scheible, 2010; Bakhshandeh and Allen, 2015):

- (2.19)
- a. “*Ed is [more old] than middle-aged*”
 - b. “*The [older] he gets, the [more cynical] he becomes.*”
 - c. “*I’d rather live in an [outer] suburb.*”
 - d. “*I’ll phone you [as soon as] the meeting is over.*”
 - e. “*my [biggest] complaint is the battery life or lack there of.*”

Sentence 2.19a is a *metalinguistic comparison*, instead of comparing “Ed” to another entity, the relative applicability of the two expressions “old” and “midlde-aged” with regard to the entity is discussed. Sentence 2.19b is a *correlative comparison* which indicates a parallel or proportional increase along the two scales “old” and “cynical”. The suffix “-er” in Sentence 2.19c is a derivational suffix (instead of an inflectional one) which converts the preposition “out” into the adjective “outer” which can be used in an attributive position. Finally, in Sentence 2.19d the phrase “as soon as” is an *idiom* meaning “immediately”. For a more in-depth discussion of these types, please refer to (Huddleston, 2002). Scheible (2010) provides Sentence 2.19e as an example where a superlative form acts as an opinion word, or more concretely in this case as an intensifier to the opinion word, but not in a comparative context.

Conversely, sometimes comparisons are expressed without using what linguists would consider comparative words. Consider these examples from (Jindal and Liu, 2006b):

- (2.20) a. “The M7500 earned a score of 85, whereas Asus A3V posted a mark of 89.”
b. “Nokia, Samsung, both cell phones perform badly on heat dissipation index.”
c. “In market capital, Intel is way ahead of Amd.”

Sentence 2.20a is a *juxtaposition*, where two complementary statements are placed next to each other without comparison markers and the comparison is implied (Cuzzolin and Lehmann, 2004). Sentence 2.20b contains the universal determiner “both” which makes this a comparison of equality about the bad performance of the two phones. Finally, Sentence 2.20c is an idiomatic way of stating that “Intel” is better than “Amd”.

For the purpose of automatically identifying comparisons in sentiment analysis, any statement about the similarity or difference of two entities is considered a comparison (Jindal and Liu, 2006a; Liu, 2015). This includes a wide variety of expressions that in some way compare two objects (cf. Examples 2.20). It excludes constructions that look like comparisons but do not compare two products (cf. Examples 2.19).

Compared to other topics in sentiment analysis, there is little work on comparisons. Jindal and Liu (2006a) are the first to specifically identify comparison sentences in product reviews. They first filter sentences that contain a comparison keyword, e.g., “more”, “favor”. From the resulting potential comparison sentences, they learn class sequential rules comprised of these keywords and the POS tags of words in a window around the keyword. They use the extracted patterns as features for a Naive Bayes classifier to distinguish comparison sentences from non-comparison sentences. Similar approaches have been successfully applied to the identification of comparison sentences in Korean (Yang and Ko, 2009, 2011a,b) and Chinese (Huang et al., 2008).

When the performance of comparison sentence detection is not the focus of the work, lists of keywords and POS have been used to find comparisons (Zhang et al., 2009, 2010). Other approaches assume every sentence with two or more entities to contain a comparison (Feldman et al., 2007; Kurashima et al., 2008; Tkachenko and Lauw, 2014, 2015). In the review domain, often even the mention of any entity other than the currently reviewed product is enough to identify comparisons (Zhang et al., 2013).

2.5.2. Comparison component detection

After sentences that contain comparisons have been identified, the next task is to decide what the relevant components of a comparison are and how to extract these components. This is the part we are going to focus on in this thesis.

One sentence may contain multiple comparisons that are independent of each other. Comparisons involve at least three components: two entities that are compared and a scale that the entities are placed on. Consider the following example:

(2.21) “[The *D200*]_{comparee} is **bigger** [than]_{pivot} the [*D80*]_{standard}.”

Of the two entities, the item that is compared is sometimes called the *comparee*. In Sentence 2.21, the comparee is the camera “*D200*”. The object the comparee is compared to is the *standard of comparison* (Heine, 1997; Cuzzolin and Lehmann, 2004; Stassen, 2013). In the example this corresponds to “*D80*”. Comparisons can omit the standard or even both entities in context for pragmatic reasons (Staab and Hahn, 1997). Comparisons are introduced by a linguistic marker called *comparative governor* (Huddleston, 2002) or *predicate* (Heine, 1997; Cuzzolin and Lehmann, 2004; Stassen, 2013). The predicate contains a *degree marker*, e.g., “*more*”, “*-er*”, (Heine, 1997). Finally, there is a *pivot*, the word that introduces the standard of the comparison, e.g., “*than*”, “*as*” (Heine, 1997; Cuzzolin and Lehmann, 2004). Sentence 2.21 contains the predicate “*bigger*”, the degree marker suffix “*-er*”, and the pivot “*than*”.

Research in sentiment analysis on comparisons has mainly focused on comparisons of products in reviews. Most of these comparisons do not compare the items in their entirety, but in some *aspect* (see Section 2.2.5). Consider the following example:

(2.22) “[The *D200*]_{E1} has a **bigger** [*LCD*]_A than the [*D80*]_{E2}.”

From this sentence, typical systems extract the predicate “*bigger*”, the two entities “*D200*” and “*D80*”, and the aspect “*LCD*”. Most approaches distinguish two types of entities, usually numbered according to the order in which they appear in the sentence.

There are two sentiment analysis approaches that attempt to extract as much detailed information as possible about comparisons. The first is presented by Jindal and Liu

(2006b) as follow-up to their work on comparison sentence identification. They use a list of manually compiled keywords to find comparative predicates. To identify entities and aspects, they use an involved pattern mining process to obtain label sequential rules containing POS tags and word forms. A very similar approach for Korean is presented by Yang and Ko (2011a). The other approach, presented for Chinese web data by Hou and Li (2008), casts the problem of extracting predicates, entities and aspects as a role-labeling task. They use a SRL system based on a CRF with features based on POS tags, the syntactic category of constituency phrases, position relative to the predicate, lemma of the comparative predicate. In their experiments they report good results on gold parse trees, but observe a large drop in performance when they use their method on automatically parsed sentences. They split the predicate into two separate parts, the particle that expresses the comparison and what they call ‘sentiment word’, the word that distinguishes the two entities, generally an adjective or adverb.

Besides these two approaches that attempt to extract all components of a comparison, others focus solely on the relation that exists between any two found entities. The actual identification of entities and aspects is more of a preprocessing step and simple heuristics are used. The most common approach to identify entities is to use lists of product names, brands and other common terms from the domain (Feldman et al., 2007; Kurashima et al., 2008; Zhang et al., 2009, 2010; Xu et al., 2009, 2011; Zhang et al., 2013; Tkachenko and Lauw, 2014, 2015). The same approach based on a list of common terms can be used to identify aspects (Feldman et al., 2007; Xu et al., 2009, 2011; Zhang et al., 2010). An alternative is to define a few high-level aspects of interest (e.g., design, functionality) and determine for each sentence whether it relates to the specific high-level aspect. This determination can be done with classification (Tkachenko and Lauw, 2014) or clustering (Li et al., 2011). Finally, some approaches simply ignore aspects (Kurashima et al., 2008; Zhang et al., 2009, 2013).

2.5.3. Identification of comparison type and direction

A comparison expresses a relation between two entities and this relation can be of different types. Linguistics distinguishes between scalar and non-scalar comparisons. *Non-scalar comparisons* are concerned with whether the compared objects are identical. There are two possible outcomes, the compared objects can either be the same (*comparison of equality*) or different (*comparison of inequality*):

- (2.23) a. “*X is the same as Y*” (equality)
b. “*X is different from Y*” (inequality)

Alternatively, a comparison can place the two objects on a scale relative to each other (*scalar comparison*). Scalar comparisons are usually expressed by a gradable adjective or adverb, e.g., “*big*”. Adjectives or adverbs that denote absolutes, e.g., “*dead*”, or those that already designate the highest grade, e.g., “*excellent*”, cannot be used in scalar comparisons. Like in non-scalar comparisons, objects in scalar comparisons can also be either identical or different. When the compared objects are not identical, we can look at their relative placements on the scale, the comparison can either be one of *superiority* or *inferiority*³. There is also a special case when the object in question is not compared only to another object (*term comparison*), but deemed to be the best out of a whole set (*set comparison*). In English, the forms of adjectives and adverbs are constructed by using the inflectional morphological suffixes “*-er*” (comparative) and “*-est*” (superlative) or the analytic markers “*more*”, “*less*” (comparative) and “*most*”, “*least*” (superlative). As a results, we have the following possible types for scalar comparisons, given here for the examples of “*big*” and “*powerful*”, adapted from (Huddleston, 2002):

- (2.24) a. “*X is as big as Y*”, “*X is as powerful as Y*” (equality)
 b. “*X is bigger than Y*”, “*X is more powerful than Y*” (inequality, superiority, terms)
 c. “*X is less big than Y*”, “*X is less powerful than Y*” (inequality, inferiority, terms)
 d. “*X is the biggest (of all Z)*”, “*X is the most powerful (of all Z)*”
 (inequality, superiority, set comparison)
 e. “*X is the least big (of all Z)*”, “*X is the least powerful (of all Z)*”
 (inequality, inferiority, set comparison)

For automatic processing in sentiment analysis, most work follows the comparison types proposed by Jindal and Liu (2006b) and further clarified in (Liu, 2015). Instead of scalar and non-scalar comparisons, they use the categories of gradable and non-gradable comparisons as the major distinction. *Gradable comparisons* are those that place the two entities on a scale and introduce a ranking between them. They come in three types. The first type is what they call *non-equal gradable* comparisons, which includes term comparisons of both superiority and inferiority. The second type includes *superlatives*, i.e., set comparison of superiority or inferiority. The third type refers to comparisons of equality called *equatives*. The first two relations also have two subtypes that clarify the direction of the relation which they call *increasing* and *decreasing* comparatives. Liu (2015) provides the following examples:

- (2.25) a. “*Coke tastes better than Pepsi.*” (non-equal gradable, increasing)

³Also called comparisons of *majority* and *minority* (Cuzzolin and Lehmann, 2004). Sometimes in the class of comparisons is included the elative “*X is very big*”, the excessive “*X is too big*” (Heine, 1997), or the assetive “*X is big enough*” (Bakhshandeh and Allen, 2015).

- b. “Coke tastes the best among all soft drinks.” (superlative)
- c. “Coke and Pepsi taste the same.” (equative)

Non-gradable comparisons express a difference between two entities, but do not rank the entities. Again, there are three types. The first type compares two entities in a *shared aspect*. The second type states that entity X has aspect A and entity Y has a *similar aspect* B. The third type states that entity X has aspect A and entity Y *does not have aspect A*. Liu (2015) provides the following examples:

- (2.26)
- a. “Coke tastes differently from Pepsi.” (shared aspect “taste”)
 - b. “Desktop PCs use external speakers but laptops use internal speakers.”
(similar aspects “external speakers” and “internal speakers”)
 - c. “Nokia phones come with earphones, but iPhones do not.” (aspect “earphones”)

In sentiment analysis research, the automatic classification of comparison types and direction has received more attention than the previous two tasks. Most work on the topic has focused on gradable comparisons only.

Jindal and Liu (2006b) introduce the system of comparison types and are the first to present a system that assigns each gradable comparison a type out of **non-equal gradable**, **equative**, and **superlative**. They use a Naive Bayes classifier with comparison keywords as features. The classification of the direction of the comparison is presented in follow-up work by Ganapathibhotla and Liu (2008). They use hand-crafted rules based on the polarity of the predicate to determine which entity is preferred in non-equal gradable and superlative sentences. Yang and Ko (2011a) adapt the same method for Korean, but they add the labels **similarity**, **pseudo-comparison** (i.e., metalinguistic comparison), and **implicit** (i.e., juxtaposition), for a total of seven classes. They do not determine the direction of the comparison.

In their work on Chinese, Hou and Li (2008) do not assign comparison types, but distinguish five possible relation types between entity 1 and entity 2 which roughly correspond to Jindal and Liu (2006b)’s comparison types: **better**, **worse**, **same**, **best**, and **worst**. The assignment of a type is based on the tokens extracted for their two-part predicates (predicate and ‘sentiment word’). They do not elaborate, but the processing seems to use manual categories assigned to the extracted words.

The above approaches assign the type and/or direction of a comparison as part of collecting detailed information about comparisons. Other research has been done in a slightly different framework where pairs of entities are extracted and the focus is on determining which of the entities is the preferred one. In this framework, there will always be two entities, whereas in the above approaches one or more of the entities may be implicit. For example the sentence “*It is the best*”, where the second entity is

implicit, will be considered to be a comparison for the above approaches, but not for the approaches that assume two entities.

A baseline method for direction identification looks up the comparison word in a sentiment dictionary (Kurashima et al., 2008; Zhang et al., 2009, 2010). If the word has positive polarity, the first entity is preferred, otherwise the second. To capture contextual influences, a polarity classifier can be used on the sentence (Zhang et al., 2013). Alternatively, if labeled data is available, a classifier can be trained to distinguish comparison directions into the classes `similar`, `different`, and `neither` (Feldman et al., 2007). None of these works directly evaluate the performance of their direction classification.

Xu et al. (2009, 2011) classify the comparative relation between two entities into four classes: three types of relations between two entities (`better`, `worse`, `same`), plus `no_comparison` which can be assigned if there is no relation between the entities. Xu et al. (2009) use a multiclass SVM and a maximum entropy model. Their features include manually defined comparison keywords, POS tags, the entity tokens and entity types. Xu et al. (2011) improve on their previous work by using a CRF and additional features from token form (capitalization, numbers, prefixes and suffixes) and dependency parses (syntactic paths between the entities, grammatical roles).

Tkachenko and Lauw (2014) identify which of the two entities in a relation is preferred. They use a generative model based on Gibbs sampling, their features are based on the position of words around the entities plus some negation treatment. In follow-up work, Tkachenko and Lauw (2015) only classify whether a relation exists or not, not the direction of the relation. They propose a dependency tree kernel for SVM that allow ‘skip-nodes’ where a node in the tree may be removed or replaced with a general placeholder which makes the approach better suited to capture similarities between dependency trees. They compare their approach to Jindal and Liu (2006a)’s system and report considerable improvements.

2.6. Summary

In this chapter, we have given an introduction to the area of sentiment analysis and presented related work. The chapter introduced basic concepts in sentiment analysis and described some approaches for the automatic detection of sentiment with a focus on those approaches that use structural context. We then covered in more detail relevant work concerning the two examples of complex verbalizations of opinions we investigate in this thesis: negation and comparisons.

3. Polarity reversing constructions

The general approach presented in this chapter and the sentence-level polarity classification experiments have been published in (Kessler and Schütze, 2012).

3.1. Introduction

This chapter discusses work on our first example of a complex verbalization of opinions: negation or more generally polarity reversers. Polarity reversers are arguably the most intuitively plausible and most frequently addressed challenge for determining the sentiment polarity of a text snippet.

Many approaches to sentiment analysis are based on sentiment words which are collected in a sentiment dictionary together with their prior polarity (positive, negative), e.g., “*good*” has positive polarity, “*disappointment*” negative polarity. These words are then used to determine the polarity of a text snippet by looking up the polarities in the dictionary and assigning the majority polarity. In some sense even machine learning classifiers that use unigram features implicitly rely on sentiment words, the prior polarity is assigned by the features weights for one class or the other (Pang et al., 2002).

While there are cases where individual words are sufficient to determine sentiment polarity, sentiment is highly compositional and context may influence and even reverse the polarity of individual sentiment words. Consider the following sentences about a digital camera from our data (the sentiment word is marked in bold with its prior polarity as subscript, the sentence polarity is given in parenthesis after the sentence):

- (3.1) a. “**good**_{pos} for wide angle or medium telephoto.” (positive)
b. “not so **good**_{pos} for wide angle or medium telephoto.” (negative)

While the sentiment word “*good*” has a positive prior polarity, in the phrase “*not so good*” the contextual polarity is negative due to the presence of the negation “*not*”. This effect is not limited to sentiment words of one polarity, a negation usually works in both directions, e.g., the phrase “*not bad*” is positive although the sentiment word “*bad*” itself has negative prior polarity.

Negation has received attention in the sentiment community since the early days of the field (Das and Chen, 2001; Polanyi and Zaenen, 2004), see also our more in-depth discussion of related work in Section 2.4. The majority of approaches has treated negation as polarity reversal, a negation changes the polarity from positive to negative or vice versa (Polanyi and Zaenen, 2004; Choi and Cardie, 2008). If polarity is given as values on a continuous scale, it is not reasonable to assume that the polarity strength of an expression is the same when it is negated, e.g., while “*fantastic*” expressed strong positive sentiment, “*not fantastic*” does not express particularly strong negative sentiment. This is why some later approaches introduce a shift negation, where a negation does not reverse polarity, but only shifts the value it to the opposite polarity by a certain factor (Taboada et al., 2011; Liu and Seneff, 2009). In this thesis, we work with only two discrete polarity values, so we treat negation as having the effect of reversing the polarity. We refer to the word or phrase that is causing the change as a *polarity reverser*. An additional point to consider is the effect of a double negation. While it can in some cases emphasize or weaken the effect of the negation (Blanco and Moldovan, 2011), we follow the standard approach in sentiment analysis and consider each negation separately, i.e., two negations cancel each other out (Choi and Cardie, 2008).

The vast majority of approaches that use sentiment words and include some treatment of polarity reversers, simply reverse a word’s polarity if it is preceded by a keyword out of a fixed list of polarity reversers (Polanyi and Zaenen, 2004; Taboada et al., 2011). Using such manually constructed fixed word lists for treating polarity reversers has several limitations, one of them being coverage, as polarity reversing expressions are diverse and can be of nearly all parts of speech (Wiegand et al., 2010). As an illustration, these are some examples from our data (polarity reversers are underlined):

- (3.2) a. “*i’ve had no **problems**_{neg} at all so far.” (positive)
 b. “*i find the lack of **entertaining**_{pos} games on this phone quite disturbing.” (negative)
 c. “*the standard battery will allow me to take pictures all day without **worrying**_{neg} about charging.” (positive)
 d. “*however, it has failed to **deliver**_{pos} on **quality**_{pos}.” (negative)****

Additionally, polarity reversal is a syntactic phenomenon which cannot be addressed solely at the word level (Wiegand et al., 2010). For efficiently using polarity reversers knowing its scope is necessary. Consider the following example:

- (3.3) “*it is not that **big**_{neg}, it **comfortably**_{pos} fits in my pocket , ...*” (positive)

Even though “*not*” acts as a reverser for “*big*”, it does not reverse the positive sentiment of “*comfortably*”. The word “*comfortably*” is outside the scope of the negation, even though the surface token distance is small.

Considering the limitations of using a manually constructed list of polarity reversing words and the importance of scope modeling, we regard polarity reversers as our first example of a complex verbalization of opinions. Specifically, our approach starts by looking at sentiment words in context and attempts to determine whether the word in question is in the scope of a polarity reverser (Ikeda et al., 2008). We call such a sentiment word’s contextual polarity *inconsistent* with its prior polarity¹. As an example consider Sentence 3.1a. The contextual polarity of the sentiment word “good” is consistent with its prior polarity: the word “good” is positive and the sentence it is contained in also expresses a positive sentiment. In the second example, Sentence 3.1b, the sentiment word “good” is inconsistent with its positive prior polarity, because the negation “not” reverses the polarity of “good” so that the final sentiment expressed in the sentence is negative, not positive as the prior polarity of the word would indicate.

Our task is then a problem of *consistency classification*, i.e., for a given sentiment word in context we want to determine whether it is consistent or inconsistent with its prior polarity. In the following polarity classification of a text based on the found sentiment words, the inconsistent words can then be treated differently from consistent sentiment words to get a more reliable result. Previous work has addressed the task of consistency classification with features on the word level, using a window-based context around the sentiment word (Ikeda et al., 2008). As polarity reversal is a syntactic phenomenon, we hypothesize that structural linguistic context information from syntax, specifically information from a dependency parse tree, is helpful for distinguishing consistent from inconsistent sentiment words. The main research question we are going to address in this chapter is thus an adapted version of research question A:

Research Question A1: *How can structural linguistic context information based on paths through a dependency parse tree be used for the reliable detection of inconsistent sentiment words in context?*

In this chapter we present a supervised machine learning approach to the problem of inconsistency classification. We compare the previously used window-based context with structural context features that represent paths through the dependency tree (which we call *syntactic constructions*). Our syntactic constructions explicitly include the scope of the reverser. Besides using all linguistic contexts for the detection of inconsistent words, we attempt to specifically identify the contexts that reverse polarity (*polarity*

¹Note that our terminology follows Ikeda et al. (2008) and differs from that used by Dragut et al. (2012) who use the term “inconsistent” to refer to a word that has conflicting polarity information in a sentiment dictionary or across dictionaries.

reversing constructions, PRCs). We present first steps towards automatically extracting PRCs from sentences annotated with polarity. In contrast to a domain-specific classifier, such constructions are more universal and can be used as a resource for consistency classification in all domains. Besides using PRCs instead of sentiment words to determine directly whether a word is consistent or inconsistent, we can also use them as features in a consistency classifier.

To summarize, this chapter presents the following points related to polarity reversal, our first example of complex verbalizations of opinions:

- We present a supervised machine learning approach to the problem of consistency classification (determining whether a sentiment word is consistent or inconsistent with its prior polarity in a given context).
- We use syntactic constructions which are paths based on a dependency tree as features for consistency classification.
- We present first steps towards automatically extracting the actual polarity reversing constructions (PRCs) from sentences annotated with polarity.
- We use PRCs directly in rule-based consistency classification.
- We use PRCs in consistency classification as features for a machine learning classifier for consistency classification.

3.2. Methods

In this section, we first discuss how to use the (in)consistency of sentiment words with their prior polarity for sentence-level polarity classification, which is the final task we want to improve (Section 3.2.1). Following this, we introduce syntactic constructions which we use to represent the scope and the syntactic context of a word based on a dependency tree (Section 3.2.2). To determine whether a word is consistent or inconsistent, we use a consistency classifier. The training of our consistency classifier requires training examples for consistent and inconsistent words. As we do not always have manually annotated data, Section 3.2.3 discusses the automatic creation of training examples from sentences annotated with polarity.

In the second part of this section, we turn to the extraction of Polarity Reversing Constructions (PRCs). From the set of all syntactic constructions, we want to identify those constructions that reverse polarities. We present our general approach of automatically extracting PRCs (see Section 3.2.4) and finally present several filtering methods that can be used to improve the extraction (Section 3.2.5).

3.2.1. Polarity and consistency classification

The final task we want to improve is *sentence-level polarity classification*. To determine sentence polarity, we follow the standard term counting approach that was first introduced by Pang et al. (2002) and has been widely used as a baseline in sentiment analysis since. This STANDARD voting approach determines the polarity of a sentence S by a “vote” from the words in the sentence. Meaning, we count the number of positive and negative words in the sentence and assign the polarity that gets the majority. In case of a tie, we default to the polarity that is more frequent in the data (usually positive for reviews). More formally, we calculate a positivity score $s_{\text{pos}}(S)$ for the given sentence S using a dictionary of positive and negative sentiment words (p and n) as follows:

$$s_{\text{pos}}(S) = |\{w \in p\}| - |\{w \in n\}| \quad (3.1)$$

for all $w \in S$. The sentence is labeled as **positive** if it contains as many or more positive words than negative words ($s_{\text{pos}}(S) \geq 0$), otherwise it is labeled as **negative**.

Standard voting counts every occurrence of a sentiment word with its prior polarity as specified in the dictionary, regardless of the context. As polarity reversal is a common phenomenon, this is clearly not sufficient for reliable polarity classification. To improve upon the standard approach, we need to determine for every found sentiment word, whether the word’s contextual polarity is the same as the prior polarity or whether it has been reversed by the context. We call this step *consistency classification*. Consistency classification assigns a score $s_{\text{cons}}(w)$ to each sentiment word w in context. Based on the resulting score, we consider w to be **consistent** ($s_{\text{cons}}(w) \geq 0$) or **inconsistent** ($s_{\text{cons}}(w) < 0$) with its dictionary polarity. The absolute value $|s_{\text{cons}}(w)|$ indicates the classification confidence.

We integrate consistency classification into polarity classification by counting a word not simply with its prior polarity, but with its contextual polarity as given by its prior polarity ($w \in p$ or $w \in n$) in combination with its consistency score $s_{\text{cons}}(w)$. Thus, we define $s_{\text{pos}}(S)$ as follows (Ikeda et al., 2008):

$$s_{\text{pos}}(S) = \sum_{w \in p} s_{\text{cons}}(w) - \sum_{w \in n} s_{\text{cons}}(w) \quad (3.2)$$

for all $w \in S$. Words expressing positive sentiment (consistent positive sentiment words as well as inconsistent negative sentiment words where $s_{\text{cons}} < 0$) add a positive value to the score, while words expressing negative sentiment (inconsistent positive sentiment

words as well as consistent negative sentiment words) add a negative value to the score.

The question we are addressing in the following is how best to determine $s_{\text{cons}}(w)$. There are two different approaches, a rule-based approach (*negation voting*) and an approach based on machine learning (*classifier voting*). Rule-based negation voting uses negation cues to determine $s_{\text{cons}}(w)$. The basic method sets the score $s_{\text{cons}}(w) = -1$ (**inconsistent**) iff an odd number of negation cues occurs in the context of w . Otherwise $s_{\text{cons}}(w) = 1$ (**consistent**). In classifier voting, a statistical classifier is used to determine $s_{\text{cons}}(w)$ and we use its classification confidence as score. In both cases, we show that representations that include sentence context as given by a dependency tree improve performance for both consistency and polarity classification.

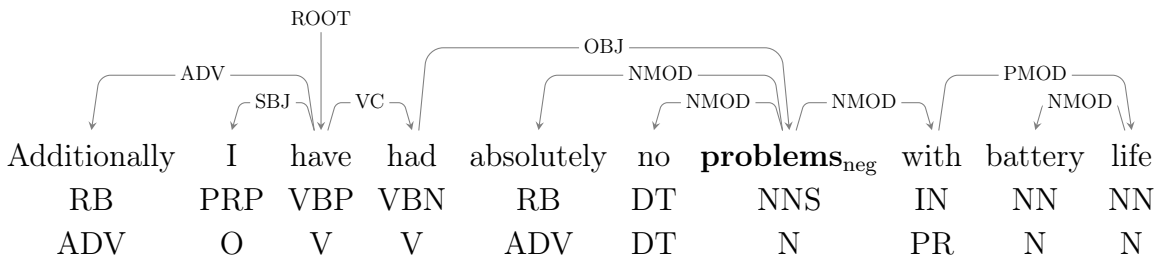
3.2.2. Representation of context as syntactic constructions

This section presents our representation of syntactic context which we call *syntactic constructions*. The simplest representation uses only the LEMMA and part of speech (POS) tag of a word. This is a baseline to compare with previous work that ignores syntax information.

To integrate syntactic information, we parse all training examples with a dependency parser. In our representations of syntactic constructions, the POS produced by the parser are generalized to the categories N (noun), V (verb), ADJ (adjective), ADV (adverb), PR (preposition), DT (determiner), and O (other, everything else). These categories correspond to universal POS (Petrov et al., 2012) with conjunctions, pronouns, number, particles and punctuation mapped to “other”.

Figure 3.1a shows the dependency parse of an example sentence. We extract *syntactic constructions* from the parses that describe the syntactic context of a sentiment word w . A syntactic construction always starts at the sentiment word w and ends at another word x in the sentence. The sentiment word is represented by its POS tag. The word itself is not included, as we are interested in constructions that are independent of specific sentiment words. The word x is represented by lemma and POS tag. We experiment with different representations for the path between the sentiment word and x .

The SIMPLE PATHS representation looks at the position of the candidate word in the dependency tree in relation to the sentiment word. There are three possible positions. The candidate word can be a child (>>) of the sentiment word, either a direct dependent or a dependent of other words that depend on the sentiment word. Alternatively, the candidate can be a parent (<<), a word on the path from the sentiment word to the root of the dependency tree. As the third possibility, the word can be a sibling (=), a



- (a) Dependency parse. The negative sentiment word “*problems*” is marked in bold. The first row below the tokens shows the POS tags assigned by the MATE parser, the second row shows the generalized POS tags we use for the syntactic constructions.

	path length $k = 1$	$k = 2$	$k = 3$
LEMMAS	absolutely_ADV no_DT with_PR have_V	life_N have_V	battery_N additionally_ADV i_0
SIMPLE PATHS	N>>absolutely_ADV N>>no_DT N>>with_PR N<<have_V	N>>life_N N<<have_V	N>>battery_N N=additionally_ADV N=i_0
ABSTRACT PATHS	N>absolutely_ADV N>no_DT N>with_PR N<have_V	N>PR>life_N N<V<have_V	N>PR>N>battery_N N<V<V>additionally_ADV N<V<V>i_0

- (b) Extracted syntactic constructions for the sentiment word “*problems*” in the different representations and different path lengths k .

Figure 3.1.: Dependency parse and extracted syntactic constructions for the example sentence “*Additionally I have had absolutely no problems with battery life.*”

dependent of an ancestor of the word, i.e., the sentiment word and the candidate share an ancestor (not necessarily the direct parent). In the example, “*battery*” is a child, “*have*” is a parent and “*additionally*” is a sibling. A simple path includes the POS of the sentiment word, the position, and the lemma and POS of the reverser candidate. The full representation of the word “*additionally*” from the example sentence with SIMPLE PATHS is NN=additionally_ADV.

Instead of reducing the relation of a candidate word and the sentiment word to one position, we can look at the complete path between the two words. The ABSTRACT PATHS representation uses all nodes on the path through the dependency tree from the sentiment word to the candidate. The words on the path are represented by their POS and the representation also contains the direction in the parse tree that needs to be traveled to come from one word to the next (up < or down >). Alternative representations could include the actual words on the path instead or in addition to the POS tags, but this would increase the sparsity of the data. For the same reason, though the label of the dependency relation could be included in the representation, we currently ignore it. Also there is some correlation between POS and dependency labels, so that it is questionable whether the additional sparsity merits the potential gain in useful information. The full representation of “*additionally*” with ABSTRACT PATHS is N<V<V>additionally_ADV.

Syntactic constructions have a parse tree distance k to the sentiment word w at the start of the construction, which is defined in number of nodes on the path. PRCs for the example sentence for distances up to $k = 3$ in all representations are given in Figure 3.1b.

3.2.3. Generation of training examples

For training our consistency classifier we need a set of training examples annotated for (in)consistency. To be independent of manually annotated data which is hard to get in large amounts, we only assume that we have a corpus of polarity annotated sentences and a dictionary of positive and negative sentiment words at our disposal. This is a reasonable assumption as several such resources have been published for English (Hu and Liu, 2004; Ding et al., 2008; Pang and Lee, 2005).

We follow Ikeda et al. (2008) and extract training examples automatically from the corpus. Given a sentiment word w with dictionary polarity p_w that appears in a sentence s with polarity p_s , we label w **consistent** iff $p_w = p_s$, and **inconsistent** otherwise. We ignore words and sentences with any label other than **positive** and **negative**. We also require the POS of the word to match the one in the dictionary.

Figure 3.2 illustrates the generation of training examples for an example sentence

as part of the whole process. Consider this example sentence from our data (original sentence from a user review including all errors in spelling and grammar):

(3.4) “The phone isn’t **hard**_{neg} to use so its **great**_{pos}” (positive)

This sentence is labeled as **positive** and contains two sentiment words, “hard” and “great”. We extract one training example per word, so the negative sentiment word “hard” is extracted as an **inconsistent** training example and the positive sentiment word “great” as a **consistent** training example. The syntactic context around the training examples is represented as described in Section 3.2.2.

3.2.4. Extraction of polarity reversing constructions

As a second task besides polarity classification, we turn to the extraction of *polarity reversing constructions* (PRCs). We define a PRC as a syntactic construction that reverses the polarity of the sentiment word in its scope, i.e., the word at the start of the construction. Our goal is the automatic extraction of PRCs. We work on the assumption that in the syntactic context of inconsistent words there is always a PRC present. Syntactic constructions that appear often in the context of inconsistent words are likely to be PRCs. There are other reasons that may create the impression of inconsistent sentiment words, e.g., sarcasm or cases where the word does not carry any sentiment, but we trust that polarity reversal accounts for the majority of cases.

We start with a set of training examples for consistent and inconsistent words (manually labeled or extracted by the process described in the previous section). All training examples are parsed with a dependency parser. Syntactic constructions are extracted from the contexts, all of them are considered candidates for PRCs. After all candidates for PRCs have been extracted, they are scored and the top n candidates are extracted as PRCs. Figure 3.2 shows the outline of our approach and an example.

The scoring of PRCs is based on comparing the number of occurrences of a candidate in the contexts of consistent and inconsistent words. If the candidate occurs with the same frequency in both contexts, the probability is low that this candidate is responsible for the polarity reversal. On the other hand, a candidate that only occurs in inconsistent contexts and never in consistent contexts is very likely to be a polarity reverser. The most straight-forward way of scoring PRCs is *relative frequency* $f_{\text{rel}}(x)$ which divides the number of occurrences of candidate x in inconsistent examples $f_{\text{incons}}(x)$ by number of occurrences in consistent examples $f_{\text{cons}}(x)$. Thus, $f_{\text{rel}}(x)$ is calculated as

$$f_{\text{rel}}(x) = \log_2 \frac{f_{\text{incons}}(x)}{f_{\text{cons}}(x)} \quad (3.3)$$

A more sophisticated method that follows the same idea is *mutual information* (MI). MI measures how much information the presence or absence of a candidate x contributes to making the correct classification decision for a sentiment word. $\text{MI}(x, C)$ between candidate x and the classes $C = \{\text{consistent}, \text{inconsistent}\}$ is defined as

$$\text{MI}(x, C) = \sum_{c \in C} P(x, c) \log_2 \frac{P(x, c)}{P(x) \cdot P(c)} + \sum_{c \in C} P(\bar{x}, c) \log_2 \frac{P(\bar{x}, c)}{P(\bar{x}) \cdot P(c)} \quad (3.4)$$

where $P(x)$ is the probability that the candidate x occurred, and $P(\bar{x})$ the probability that x did not occur.

While MI extracts candidates that serve as a good indicator for one of the classes, these candidates are not necessarily good indicators for the class **inconsistent**. To get only indicators for the class **inconsistent**, we place the candidates that are indicators for the other class at the end of the list when we sort by MI. We call this measure *extended mutual information*, MI^+ . As MI is always a positive number, MI^+ is calculated as

$$\text{MI}^+(x, C) = \begin{cases} -\text{MI}(x, C) & \text{if } P(x, \text{consistent}) > P(x, \text{inconsistent}), \\ \text{MI}(x, C) & \text{else.} \end{cases} \quad (3.5)$$

In cases where we have more candidates than the number of PRCs we want to extract, the above is equivalent to removing candidates with negative association, i.e., candidates where $\log(P(x, \text{inconsistent})/P(x, \text{consistent})) < 0$, from the final set of extracted PRCs (Dunning, 1993).

3.2.5. Filtering of generated training examples

The set of automatically extracted PRCs is rather noisy. Part of the reason is that the training examples for (in)consistent words are automatically extracted and contain a considerable amount of noise. We experiment with different filtering methods to create a cleaner set of training examples and resulting PRCs. These filters are based on an analysis of wrongly extracted examples, but address general issues that are common in the area of sentiment analysis. Our filters are designed for high precision on filtering out incorrect examples rather than recall.

The first and most important issue affecting the extraction of training examples is subjectivity (cf. Section 2.2.1). Sentiment words do not express sentiment in every given context. In a preliminary study on a subset of sentiment words extracted as training examples, we found that about 50% of words extracted as inconsistent training examples

did in fact not express sentiment in the sentence context. Consider the following example with the marked sentiment words:

(3.5) “*easy*_{pos} to hold steady when using *slower*_{neg} shutter speeds.” (positive)

The word “*slower*” does not express sentiment in this context, rather it describes a setting. Identifying and discarding non-subjective phrases like these would improve the classification results as well as the quality of the extracted polarity reversing constructions. We implement several filters to tackle this issue.

Different words have different degrees of subjective usages². In one of the dictionaries we are using, two categories of sentiment words were distinguished and manually annotated: *weakly* and *strongly* subjective words. Strongly subjective words are subjective in most contexts, whereas weakly subjective words only have certain subjective usages but may have other non-subjective uses (Wilson et al., 2005). To get a higher percentage of actually subjective usages in the training examples, we use only strongly subjective sentiment words from the dictionary. Some examples of strongly subjective words include “*excellent*”, “*great*”, “*disappointment*”. Examples for weakly subjective words are “*slow*”, “*easy*”, “*lose*”. We call this filter SUBJSTRENGTH.

Some sentiment words from the dictionary can occur with a POS that never expresses sentiment such as interjection. Consider these examples of extracted training examples for the positive sentiment word “*okay*”:

- (3.6) a. “*pictures are okay*_{pos,JJ} but not great.” (negative)
 b. “*Okay*_{pos,UH}, now to the cons: The battery life could definitely be better.” (negative)

In the first example, the word occurs as an adjective (JJ) and expresses an opinion. In the second example it occurs as an interjection (UH) to introduce a new part of the review and does not express any opinion, so no training example should be extracted. There are several such POS that never express sentiment which we ignore: interjection (UH), preposition (IN), determiner (DT) or cardinal number (CD). Note that we are not removing the words from the dictionary, we only ignore occurrences in the data with these specific POS. We call this filter POSIGNORE.

Another reason that a sentiment word may not express an opinion is that sometimes sentiment words occur in an objective context as part of aspects or environment descriptions. A few examples from our data:

- (3.7) a. “Very minor chromatic distortion with *wide*_{pos} angle ...” (negative)
 b. “in *low*_{neg} *light*_{pos} conditions images tend to get blurred.” (negative)

²“reliability” in Wilson et al. (2005)

POS ₀	POS ₁	POS ₂	Example aspects
	NN	NN	“battery life”, “picture quality”
(CD)	JJ	NN	“optical zoom”, “mega pixel”
(JJ)	JJ	NN	“sound quality”, “wide angle”
(IN)	JJ	NN	“bright sunlight”, “low light”

Figure 3.3.: Patterns used for aspect bigram extraction. The token matching POS₀ is not extracted as part of the bigram.

- c. “The compact **flash**_{neg} card is easy to use and seems more durable to me that the **smart**_{pos} media card.” (positive)

While a camera having a “wide angle” expresses positive sentiment about the camera, in this case the “wide angle” only refers to a setting for the camera. Other similar cases are environment descriptions (“in bright light”) or names of camera parts or accessories (“flash card”, “smart media card”).

This can be viewed as sentiment words occurring as part of a product aspect³ which should be ignored. As we do not have a list of aspects from our domain, we extract aspects automatically. Following Glaser and Schütze (2012), aspects are most often nouns or contain a noun. Single noun aspects are a problem of the dictionary, at this point we are interested in bigram aspects. To extract these, we define a list of extraction patterns on POS as shown in Figure 3.3. In trigram patterns only the last two tokens are extracted as a bigram, but the POS of the word before the extracted bigram has to match the first POS of the pattern. We extract the most frequent bigrams as aspects for our domain. A manual inspection of the extracted bigrams confirms the claim of Glaser and Schütze (2012) that nearly all frequent bigrams in a domain corpus are aspects, so no further processing is performed. Once created the list of aspects, we ignore sentiment words that occur as part of an aspect in the text. We call this filter ASPECTS.

Finally, another reason that a sentiment word does not express sentiment is that it is acting as an intensifier for another sentiment word. Consider the following examples:

- (3.8) a. “It is **ridiculously**_{neg} **easy**_{pos} to use.” (positive)
 b. “digital zoom is **pretty**_{pos} **useless**_{neg} because of quality.” (negative)
 c. “The sound quality is **vastly**_{pos} **improved**_{pos}.” (positive)

Even though “ridiculous” on its own is a negative sentiment word, in the example it only serves to intensify the positive sentiment of “easy” and should not be extracted

³While settings and parts certainly qualify as product aspects, environment descriptions would probably not qualify, but as they follow the same patterns, we include them with aspects for our purposes.

as a reversed training example. For our filter, we ignore adverbs that modify another sentiment word (of all POS tags, these will be adjectives or verbs). Adjectives can also intensify nouns, but the situation is less clear, so we currently restrict ourselves to adverbs. We call this filter INTENSIFIER.

Besides words that do not express sentiment, we have a second issue that produces errors in the extraction of training examples: words that express a different polarity than the one contained in the dictionary. The domain-specificity of sentiment words is well-recognized in sentiment analysis (Pang and Lee, 2008). The sentiment dictionary we are using has been created for the domain of news texts, but we are analyzing product reviews. Consider the following examples:

- (3.9) a. “*photos are **sharp**_{neg} with acurate color.*” (positive)
 b. “*technical **support**_{pos} for both samsungsprint pcs are in the dark.*” (negative)
 c. “*you have the options of taking **black**_{neg} and white photographs and there are some color filters available as well.*” (positive)

The word “*sharp*” is contained in the dictionary as a negative sentiment word, but the usage in our domain in the context of pictures is always positive. Another example is “*support*” which the dictionary assumes to be a positive sentiment word, but most occurrences in our data are neutral, e.g., “*technical support*” or “*customer support*”.

We want to remove words that are not sentiment words in our domain or mostly express a different polarity, but we do not have a domain-specific list and do not want to invest the manual annotation effort. As we have lots of annotated data with sentiment on the document-level, we can train a classifier with unigram features in the domain. The unigram features with high weight for a class are good indicators for the class in the domain. But not all features are sentiment words, e.g. “*pda*” has high weight for the positive class in our domain, because it is something special and good for a phone to be usable as a PDA (personal digital assistant)⁴. Using the classifier features directly as sentiment words would not give very good results. We therefor use the classifier features only to filter the manually created sentiment dictionary. Words from the subjectivity clues list are only included in our list of sentiment words if they are among the *c* features with highest weight for the class. We call this filter DOMAIN.

Finally, instead of filtering the training examples or the dictionary, we can also filter the resulting set of PRCs. One case that creates false positive PRCs is a fixed word combination where one of the word is a sentiment word and the combination often occurs in the combination in sentences with the opposite polarity. An example is the

⁴The reviews in our data are from the years 2006–2008 where this was a thing.

construction ADJ<drive_N which is extracted as a PRC, because “hard” is a negative sentiment word and the combination “hard drive” often occurs in positive contexts:

- (3.10) a. “The main reasons I purchased this device were [...] **hard**_{neg} drive, [...]” (positive)
 b. “It acts like a small **hard**_{neg} drive!” (positive)

All instances of the construction ADJ<drive_N are in the context of mentioning “hard drive” in a positive context, there is no other adjective that is used in this combination with “drive” in our data. To avoid PRCs that do not generalize to other sentiment words, we exclude candidate paths that are always extracted from the same sentiment word in the corpus. We call this filter SINGLETONS.

To summarize, our filters address

- subjectivity (filtering training examples for non-sentiment uses of sentiment words: SUBJSTRENGTH, POSIGNORE, ASPECTS);
- intensifiers (filtering training examples for uses as sentiment INTENSIFIER);
- domain-specificity (filtering the dictionary for systematic polarity differences in the DOMAIN); and
- spurious PRCs (filtering the resulting PRCs for SINGLETONS).

3.3. Automatic polarity annotation of sentences

Our approach for extracting PRCs requires a large number of labeled sentences to get enough training examples for a reliable extraction. The sentence-level annotated customer review data sets (Hu and Liu, 2004; Ding et al., 2008) that we use for evaluation only contain about 2000 sentences each and thus are too small for our purposes. Another sentence-annotated corpus is the sentence polarity movie review data set (Pang and Lee, 2005) of about 10000 sentences. This data set is not the result of manual annotation of sentences, but labels have been automatically assigned to snippets that are used to represent the reviews on rottentomatoes.com. Snippets may not be sentences. Also, this data comes from the movie domain, which is a very different domain from the electronics reviews we are working on. This is why we decide against using this data set.

To avoid the manual annotation effort of annotating a large number of sentences, we make use of semi-structured reviews to automatically extract labeled sentences. We use the sentences only for the automatic extraction of PRCs, not to evaluate polarity classification. We call this data set PROSCONS.

Figure 3.4 shows an example of a semi-structured review. In addition to the written text of the review and a star rating, the users provide “pros” and “cons”. The “pros” refer

"Nice camera, worth the money"

★★★★★ on February 23, 2009 by [jmos1](#)

Pros

Easy to use, very good outdoor pics, battery life is very good.

Cons

Indoor flash pics tend to be too bright for close objects.

Summary

This is my 5 digital camera and my first DSLR. Overall I am very pleased with the photo quality and easy of use. Color balance has been right on for most pics. Battery life has been exceptional. Having two manual control knobs rather than having to use the menus has been very positive and fast.



[Reply to this review](#) | Was this review helpful? 0  0  0



Figure 3.4.: Example of a semi-structured review from [epinions.com](#) (screenshot taken on June 13th, 2012).

to aspects of the product the user evaluates as especially positive, the “cons” to aspects the user evaluates as especially negative. We refer to both collectively as *keyphrases*. Our goal is to align the aspects discussed in the keyphrases with the review text. Assuming that one aspect is evaluated consistently as positive or negative in one review, we can then extract sentences referring to the aspects. If the aspect was mentioned in a “pro”, the corresponding sentence receives the label **positive**, if the sentence was extracted because of an aspect mentioned in a “con”, it receives the label **negative**.

For our extraction, we distinguish three different types of keyphrases as illustrated by these examples:

- (3.11) a. “*Indoor flash pics tend to be too bright for close objects*”
 b. “*HD video*”
 c. “*Easy to use*”

Keyphrase 3.11a is a complete sentence and can be extracted as-is. We use all keyphrases longer than 3 tokens directly as a sentence with the corresponding label. Keyphrase 3.11b only states an aspect of the product. We take the complete keyphrase to be an aspect and check for every sentence in the review text whether it contains the aspect. If so, we extract complete sentence with the corresponding label. Keyphrase 3.11c names an aspect, but also contains a sentiment word. In this case we remove all sentiment words using a sentiment dictionary, and then proceed the same way as with keyphrases containing a single aspect.

	Cameras	Cellphones
Number of reviews	12586	4856
Avg. review length (words)	1014.2	1056.9
Avg. number of keyphrases	4.84	4.91
No. annotated sentences extracted, keyphrases	19995	5987
No. annotated sentences extracted, text	22143	10378
No. annotated sentences extracted, total	42138	16365

Table 3.1.: Statistics of automatically annotated PROSCONS data set.

	#Sentences	Agreement	κ
Cameras, keyphrases	312	0.88	0.77
Cameras, text	324	0.70	0.32
Cameras, total	636	0.79	0.61
Cellphones, keyphrases	305	0.89	0.77
Cellphones, text	330	0.71	0.37
Cellphones, total	635	0.80	0.62
Total	1271	0.79	0.61

Table 3.2.: Agreement and κ between the manual and automatic annotation of a random subset of sentences from PROSCONS extracted by our method.

Our process is not intended to provide a label for every sentence in every review, but instead to extract a few sentences where we can be reasonably sure about the label. Branavan et al. (2009) have researched the relation of keyphrases with the review text and note two characteristics they call incompleteness and inconsistency. Incompleteness refers to the fact that not every aspect mentioned in the keyphrases will be mentioned in the text and not every aspect mentioned in the text will have an associated keyphrase. In the subset of reviews Branavan et al. (2009) investigated, they found more than 40% of reviews to be incomplete. Inconsistency refers to the issue of using different words to refer to the same aspect, e.g., a user might use “*user-friendly*” instead of “*easy to use*” or users might talk about “*photo quality*”, “*picture quality*”, “*image quality*”, “*quality of images*” and so on. This is not only a cross-review problem, but happens even in the same review. In the domain of restaurants, Branavan et al. (2009) count 15–27 different phrases for each of the six most common properties of the domain where the most frequent phrase was used to describe the aspect in only 33% of cases on average.

Due to incompleteness, labeling every sentence in a review by using keyphrases will not be feasible, even by a perfect method. But more sophisticated methods may be

able to address sentences that our system misses because of inconsistency. We also miss aspects that are listed in a complete sentence in the keyphrases, we do not check for this aspect in the review text, though it may still appear in a sentence in the review text.

Still, even with our basic method we manage to get a sufficient number of annotated sentences for our purposes. We apply our extraction method to an existing corpus of semi-structured reviews from `epinions.com` containing camera and cellphone reviews⁵ (Branavan et al., 2009). The different aspects in pros and cons are already separated. Table 3.1 contains statistics about the data. Our extraction methods results in about 58000 annotated sentences.

To judge the quality of the extraction method, we have manually checked the labels of a random subset of the automatically annotated sentences. We had one annotator, a graduate student of computational linguistics, who annotated a total of 1271 sentences. Half of the sentences were from the cameras part and half from the cellphones part. For each part the sentences were distributed roughly so that half of the sentences were labeled keyphrases and the other half sentences extracted from the review text. Table 3.2 shows some details about the agreement numbers. The agreement of the automatic and manual annotation on the complete set is rather high with 0.79. Cohen’s κ is 0.61 which is considered substantial agreement.

The following reflects some sentences where the annotator disagreed with the system on keyphrases that were extracted as complete sentences:

- (3.12) a. “*NONE this is one of the best*” (con vs. **positive**)
 b. “*slightly more costly than others but worth it*” (con vs. **positive**)
 c. “*produces pseudo shutter sound*” (pro vs. **neutral**)

Sometimes users do not have anything to list for a pro or con and instead of just leaving the field empty, they write something that does not really refer to any aspect of the product (Sentence 3.12a). Also, they might want to hedge a con for a very positive review or play down a pro for a very negative review, like in Sentence 3.12b where the keyphrase contains more than one polarity. The annotation was done on the sentence alone, without the context of the review, so it is sometimes difficult to decide what sentiment is expressed and what it is expressed on. Also some judgments are subjective, presumably the reviewer writing Sentence 3.12c likes that the camera makes a sound while other users may disagree about that.

Agreement on sentences that are extracted from the review text based on aspects mentioned in the keyphrases is lower than for complete keyphrases, as there are more possibilities for errors. Consider the following sentences:

⁵<http://groups.csail.mit.edu/rbg/code/precis/> (camera and cellphone data sets)

- (3.13) a. *“Because they’re rather inexpensive compare to other memory options, for example, a 64MB Memory Stick would cost double the price of a 64MB CF.”*
(con “cost” vs. **neutral**)
- b. *“Canon S60 - nice wide angle, but extremely slow camera and mediocre battery life.”*
(pro “good battery life” vs. **negative**)
- c. *“The button is small and hard to push, and the camera beeps with each push.”*
(pro “small” vs. **negative**)

Sometimes an aspect may occur in a neutral sentence that says something general about some aspect (Sentence 3.13a), as the name of a setting or in a description. Longer reviews frequently compare a product to others, so an aspect that is positive for the given product may be mentioned as negative for a different product (Sentence 3.13b). Also, a more generic aspect may be matched to a different context that actually talks about a different aspect of the product (Sentence 3.13c) .

There are many directions to improve the automatic polarity annotation of sentences from semi-structured reviews, both in terms of precision and in terms of recall. Still, agreement numbers are not bad and we expect the advantage of having a large amount of data to outweigh the noise introduced by incorrect labels.

3.4. Experiments on PRC extraction

The first set of experiments evaluates the approach for automatically extracting Polarity Reversing Constructions (PRCs) as presented in Section 3.2.4 and explores the use of the filters introduced in Section 3.2.5.

3.4.1. Data and experimental setup

For our approach we need a dictionary of sentiment words, a large amount of sentences labeled with polarity and a set of manually labeled PRCs for evaluation.

Sentiment dictionary. As our sentiment dictionary we use the MPQA subjectivity clues⁶ (Wilson et al., 2005). An entry in the dictionary consists of a word, its POS tag (one of ‘noun’, ‘verb’, ‘adj’, ‘adverb’ or ‘anypos’ which are mapped our generalized POS tags N, V, ADJ, ADV and O), its polarity and a subjectivity strength (strongly or weakly subjective). The same word may have two or more entries with different polarities, with one exception (“boast” as a verb) these are for different POS tags. If

⁶http://mpqa.cs.pitt.edu/lexicons/subj_lexicon/

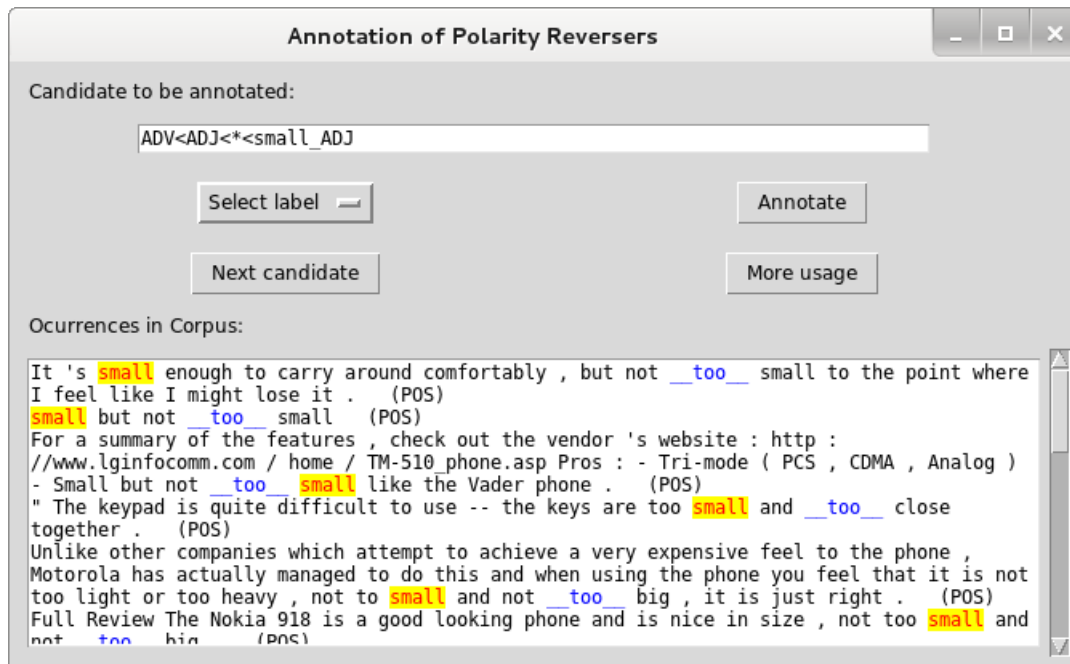


Figure 3.5.: Screenshot of the annotation tool for PRC annotation. The candidate PRC is marked in red on yellow background, the sentiment word it modifies is enclosed in `__` and marked in blue.

there are several polarities for a given word in a given context, it is always treated as consistent (majority class). The dictionary contains 6456 words; 2304 of them positive (2718 entries) and 4152 of them negative (4911 entries). We refer to this dictionary with MPQA in the following.

Labeled sentences. As data for the extraction of PRCs we use the PROSCONS sentences created as described in Section 3.3. The sentences from the camera and cellphone parts are used together, as the evaluation data is also from a mixed domain. In total, we have 58504 sentences. With the MPQA dictionary, we find a total of 83269 sentiment words in 42949 of the sentences. 24303 of the found sentiment words have a prior polarity inconsistent with the sentence polarity (29%).

Gold PRCs. To directly evaluate the extracted PRCs against a set of gold-standard PRCs, we had our annotator label some syntactic constructions as PRCs or non-PRCs. For the annotation, the annotator was presented the syntactic construction in the form of ABSTRACT PATHS along with a number of usage examples. In these examples the candidate word of the PRC and the sentiment word at the end of the path were marked. The annotator had to chose between the labels `reverser`, `nonreverser` and `uncertain`.

Uncertain examples were decided in discussion with the author. Figure 3.5 contains a screenshot of the annotation tool.

The resulting set of 70 gold PRCs can be found in Figure A.1. Note that we did not attempt to create an exhaustive list of all existing PRCs. Rather we annotated the constructions extracted as PRCs by different versions of our system to see which of them are truly PRCs.

Experimental setup. We run our PRC extraction approach presented in Section 3.2.4 for all three representations for syntactic constructions (LEMMAS, SIMPLE PATHS and ABSTRACTED PATHS) and test the three different scorings of PRCs (relative frequency f_{rel} , Mutual Information MI, extended Mutual Information MI+) with path lengths between $k = 1$ and $k = 6$. Corresponding to the number of manually annotated constructions in the gold PRCs set, we extract the top 70 constructions. To exclude infrequent outliers, we set a minimum threshold of 10 occurrences for a construction in all experiments. As we do not have an exhaustive list of all existing PRCs, we do not evaluate the recall of our approach. Rather we present results for the percentage of the top n extracted constructions that are actual PRCs, i.e., precision.

3.4.2. Results and discussion

Basic results

Figure 3.6 shows some results in terms of the percentage of the extracted constructions that are actual PRCs contained in our list of gold PRCs (the complete results are in Table A.2). For LEMMAS, adding more context by increasing k leads to less actual reversing words that are found with f_{rel} and MI+. This is understandable as most reversing words occur close to the sentiment word. Adding more words from the context without position information only adds noise. As an example consider the word “without”. At $k = 1$, it occurs more often in reversing contexts, e.g., “without **worrying**_{neg}”. Using $k = 2$ adds a large number of occurrences in non-reversing contexts, e.g., “**great**_{pos} pictures without flash”. The difference in number of occurrences is still large enough that “without” receives a high MI score, but it gets discarded with MI+ as it occurs more often in non-reversing contexts.

SIMPLE PATHS profits from the introduction of siblings at $k = 2$ which is a very frequent construction for polarity reversal in predicative sentences:

(3.14) “However, the battery is not very **good**_{pos}” (negative)

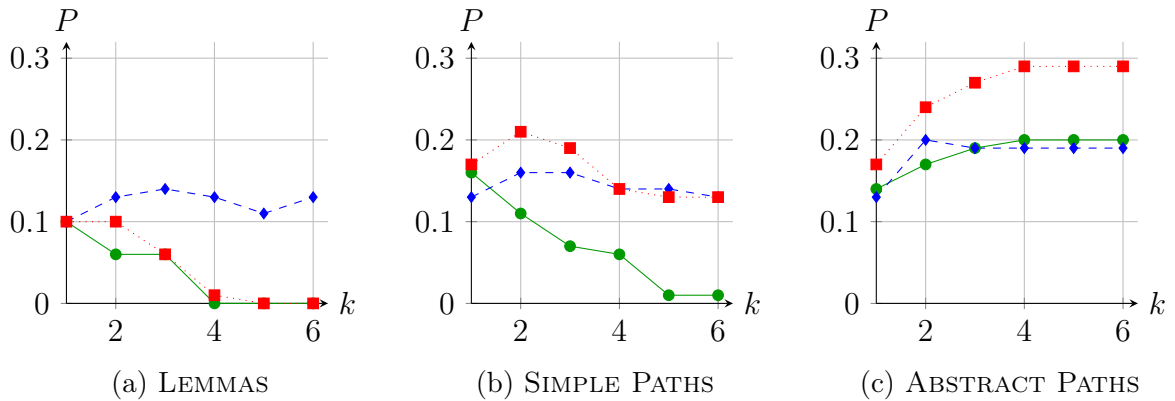


Figure 3.6.: Percentage of correctly extracted PRCs at $n = 70$ for different representations (LEMMAS, SIMPLE PATHS and ABSTRACTED PATHS), path length k between 1 and 6 and scoring with f_{rel} , MI and MI+.

Both “not” and “good” are dependents of “is”. With bigger k , occurrences further away cannot be distinguished from close occurrences and add noise, just like with LEMMAS.

For ABSTRACT PATHS increasing k has a positive effect until about $k = 4$. This makes sense as ABSTRACT PATHS explicitly includes the distance k and uses it to distinguish different constructions. After $k = 4$ the results do not change much, as only very few longer constructions occur sufficiently often to make it into the top of the list.

The highest number of actual PRCs are found when representing syntactic context with ABSTRACT PATHS, but in general, numbers are rather low. Of the top 70 ABSTRACT PATHS extracted as PRCs at $k = 4$ with MI, only 13 are correct (19%). For the same setting, but using f_{rel} only 14 are correct (20%). Results for MI+ are better, but still noisy: 20 out of 70 constructions are correct (29%).

Not only the numbers, but also the syntactic constructions extracted with the different scoring methods are very different. Figure 3.7 shows the top ten constructions extracted for ABSTRACT PATHS with each of the three scoring methods.

Out of the top 10 constructions extracted with f_{rel} , none is an actual PRC. The top construction for scoring with f_{rel} , ADJ<V<0>N>battery_N, comes from sentences like “battery life could be better” which express a negative sentiment. The related construction ADJ<V<0>life_N is on position 11. Although “could/would/should be ADJ” (ADJ<V<0>) is used in combination with other aspects as well (“lcd quality could be a bit better”, “10x zoom would be nice”, ...), the phrase “battery life” is the one most mentioned in this construction (39 times compared to 14 times for the second-most frequent word “quality”). The second-ranked construction, N<cards_N, is an example of a construction extracted because of the occurrence of sentiment words in aspects like “flash cards”.

ADJ<V<O>N>battery_N ✗	ADJ>O>use_V ✗	ADJ>not_ADV ✓
N<cards_N ✗	ADJ>to_0 ✗	ADJ<V>not_ADV ✓
ADV<ADJ<O<small_ADJ ✗	ADJ>not_ADV ✓	N>no_DT ✓
PR<V>ADJ>quite_ADV ✗	ADJ<V>not_ADV ✓	ADJ<V<could_0 ✓
ADV<N<V<that_PR ✗	ADJ<V>N>the_DT ✗	N>low_ADJ ✓
PR>V>and_0 ✗	N>no_DT ✓	N<N>low_ADJ ✗
PR<V>ADJ>O>use_V ✗	ADJ<V<could_0 ✓	N<in_PR ✗
PR<V>feature_N ✗	ADJ>very_ADV ✗	ADJ<N>not_ADV ✓
ADJ<O<ADJ>bit_N ✗	N>low_ADJ ✓	PR<be_V ✗
ADV<ADJ<but_0 ✗	ADJ<V>and_0 ✗	V<V>not_ADV ✓

(a) f_{rel} (b) MI (c) MI+

Figure 3.7.: Top 10 extracted PRCs for ABSTRACT PATHS at $k = 4$ with different scoring methods. Correct PRCs are marked with ✓, extracted non-PRCs with ✗.

With MI, some actual PRCs are found in the top 10 constructions. The two top constructions for MI, ADJ>O>use_V and ADJ>to_0, are both indicators for the non-reversed class as it is often used in constructions like “*easy to use*” in positive contexts. The negated version, “*not easy to use*”, appears to be less frequent.

In the results for scoring with MI+, the top two constructions of MI are filtered out, as they are indicators for the non-reversed class. All actual PRCs extracted by MI are kept. The first errors for MI+, N<N>low_ADJ at position 6 and N<in_PR at position 7, are examples for a description of the environment or settings, e.g., “*low light shooting*” or “*in low light*” which we have already discussed in context with filters.

Results with filters

After finding the best settings for the basic parameters, we now apply the different filters discussed in Section 3.2.5. We use the best performing system to test the filters, i.e., ABSTRACT PATHS with MI+ scoring, which is included in the plots as BL. Figure 3.8 shows some results split into two plots for better readability.

We can see that the SUBJSTRENGTH filter performs worst of all, introducing errors instead of removing them. When we apply this filter, the number of found sentiment words drops drastically from about 83000 to only about 30000. This is expected, but the expectation was that the remaining training examples are of better quality because non-subjective uses of words where there is no real reverser present are excluded. A manual inspection of the filtered words confirms that this is not always the case. Even if there is overall a slightly better quality of training examples, this cannot compensate

for the smaller amount of training examples overall.

For the DOMAIN filter, we train the Stanford MaxEnt classifier⁷ (Manning and Klein, 2003) with unigram features and default settings on the PROSCONS data to distinguish **positive** from **negative** sentences. We use the *c* features with the highest weights for each class to filter the dictionary. We exclude non-word features, which leaves about 10500 features. For illustration, the top 10 positive features from the classifier are “*capable*”, “*amazing*”, “*wonderful*”, “*convenience*”, “*tons*”, “*inexpensive*”, “*telephone*”, “*pros*”, “*excellent*”, “*solid*”. The top 10 negative features are “*lacks*”, “*worst*”, “*heats*”, “*poor*”, “*horrible*”, “*scratches*”, “*dislike*”, “*ll*” (from “*I’ll*”), “*fragile*”, “*concern*”.

The plot shows the result for 1000 features, but the results are very similar for all cases where we filter for the up to 7000 top features. For the top 1000 features, the resulting filtered MPQA dictionary contains only 131 positive and 145 negative words. As we can see, performance drops sharply, nearly as much as for SUBJSTRENGTH. Only after using the top 8000 features, the numbers rise again to the baseline, but never improve upon it. At this point the dictionary contains about 600 words of each polarity and 75000 sentiment words are found – nearly all that are contained in the data. The reason for the low performance is that many bad sentiment words are not filtered out, because even though they do not occur as sentiment words, they still occur in a context that fits their dictionary polarity. For example while the word “*brightness*” is not a positive sentiment word in our domain, it is often mentioned in positive contexts such as “*you can adjust the brightness*”, so it still gets a high weight for the class **positive** from the classifier.

Using the filter INTENSIFIER does not have a significant influence on the results. This filter only affects the extraction of PRCs that have adverbs in their scope. Constructions of length one which end at a sentiment word, like ADV<useless_ADJ or ADV<bad_ADJ, are dropped completely. Most other constructions just change position a few places up or down, because intensifying uses are found with roughly the same percentage in reversing and non-reversing contexts. As a result, the final extraction performance stays roughly the same with the filter as without using the filter.

The difference to the baseline when filtering for ASPECTS is even smaller than for INTENSIFIER. Aspects are extracted from the PROSCONS corpus. There are a total of 17484 bigram phrases, 32 occur 100 times or more, but only 200 occur more than 20 times. The plot contains the results for using the top 100 aspects. Varying the number of aspects in the list to ignore between 0 and 150 does not make a big difference in the result, afterwards results drop slightly. When we look at the two constructions

⁷<https://nlp.stanford.edu/software/classifier.html>

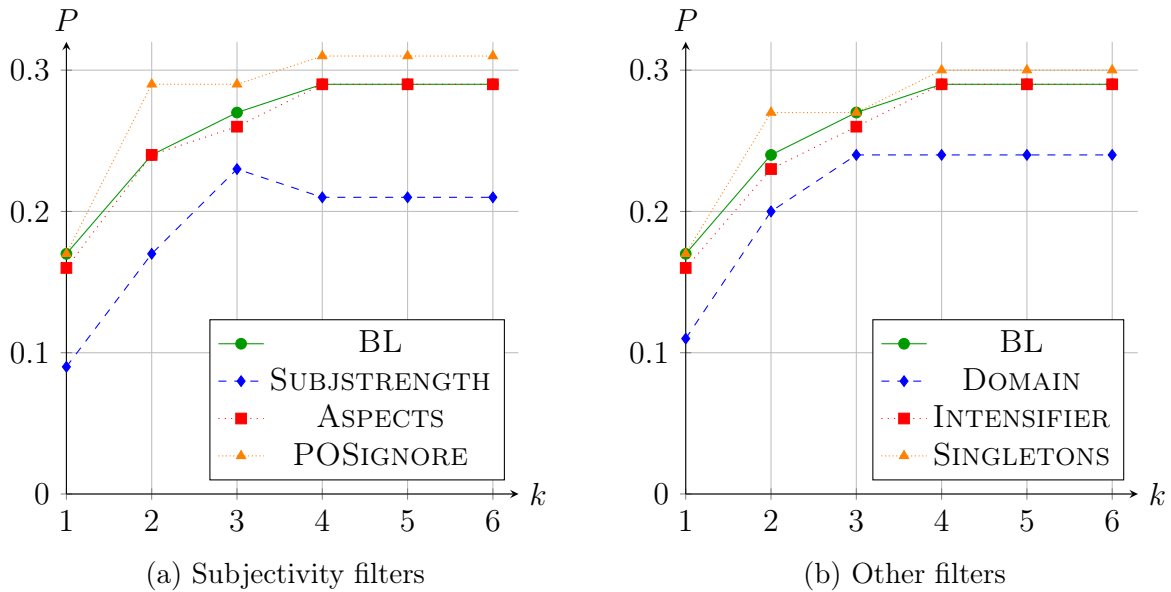


Figure 3.8.: Percentage of correctly extracted PRCs at $n = 70$ for different path lengths k with filters. BL refers to the best system of the base settings, i.e., ABSTRACT PATHS scored with MI+.

discussed as examples in Section 3.2.5, $N\langle N\rangle\text{low_ADJ}$ disappears completely from the list and $N\langle\text{in_PR}$ is relegated to position 2106. Unfortunately, they are replaced with equally bad alternatives: $\text{PR}\langle\text{be_V}$, $\text{PR}\langle\text{V}\rangle\text{N}\langle\text{the_DT}$ (mainly because of “*although the X is*” with “*although*” as a sentiment word) and $\text{ADJ}\langle\text{at_PR}$ (from “*at least*” or “*at best*”).

The filter SINGLETONS is the first to improve upon the baseline, although only slightly. The first change is to (correctly) remove $\text{ADJ}\langle\text{O}\rangle\text{at_PR}$ at position 29 which only occurs in combination with the sentiment word “*least*” when used like “*at least 130k pixels*” where “*least*” is not a sentiment word at all. Next are $\text{ADJ}\langle\text{O}\rangle\text{white_ADJ}$ (“*black and white*”) and some constructions with the sentiment word “*although*”: $\text{PR}\langle\text{V}\rangle\text{it_O}$, $\text{PR}\langle\text{V}\rangle\text{i_O}$ and $\text{PR}\langle\text{V}\rangle\text{not_ADV}$. The example that prompted the filter, $\text{ADJ}\langle\text{drive_N}$ is still in the list, because the data contains several modifiers for the word “*drive*”, some examples are “*free [zip] drive*”, “*powerful [high-speed lens] drive*”, or “*portable [hard] drive*”.

The biggest improvement over the baseline comes from using the filter POSIGNORE, although it still only improves precision from 29% to 31% for $k = 4$. POSIGNORE allows us to get rid of $\text{PR}\langle\text{be_V}$, $\text{PR}\langle\text{V}\rangle\text{N}\langle\text{the_DT}$ and many other constructions that are based on occurrences of very questionable sentiment words like “*although*”, “*at least*” or “*above*” which are almost all in non-sentiment contexts. We have tried the combination of POSIGNORE with the other filters, but got no or only marginal improvement. In all cases correctly excluded constructions were replaced with equally bad alternatives, so

that the overall performance change was minimal.

Our final set of PRCs that we use in the following experiments is the set extracted with the following settings: ABSTRACT PATHS, MI+ scoring, path length $k = 4$ and POSIGNORE filter. While even our highest result does not look very promising (only 31% of extracted constructions are really PRCs), as we will see, we can still use noisy PRCs successfully in polarity classification.

3.5. Experiments on consistency and polarity classification

The second set of experiments evaluates our system on the tasks of word-level consistency classification and sentence-level polarity classification as described in Section 3.2.1.

3.5.1. Data and experimental setup

Sentiment dictionary. As our sentiment dictionary we use the MPQA subjectivity clues described in the previous section (Wilson et al., 2005).

To investigate the influence the sentiment dictionary has on the results, we repeat the experiments with the General Inquirer dictionary (Stone et al., 1966) which contains 3518 words; 1775 words or 2083 entries positive and 1743 words or 2074 entries negative⁸. The entries consist of a word, its POS tag and its polarity. The dictionary contains the POS tags 'noun', 'modif', 'supv' which are mapped to our generalized POS tags N, ADJ, V respectively. If nothing is found O is assigned. We refer to this dictionary by GI in the following. There are 23 cases where the word with the same POS tag can have both polarities. If there are several polarities for a given word in a given context, it is always treated as consistent (majority class).

Sets of PRCs. Our final set of PRCs that we refer to as PRC-SYSTEM in the following experiments is the best set of 70 extracted PRCs from the previous set of experiments, i.e., the set represented as ABSTRACT PATHS and created with MI+ scoring, path length $k = 4$ and POSIGNORE filter. We additionally use the manually annotated list of gold PRCs (PRC-GOLD) described in the previous section extraction as an upper bound of the performance we can expect with automatically extracted PRCs.

⁸using the categories from Choi and Cardie (2008); for positive words: 'pos', 'pstv', 'posaff', 'pleasur', 'virtue', 'increas'; for negative words: 'negativ', 'ngtv', 'negaff', 'pain', 'vice', 'hostile', 'fail', 'enloss', 'wlbloss', 'tran-loss' (see <http://www.wjh.harvard.edu/~inquirer/homecat.htm>).

Evaluation data. As evaluation data we use the two customer review data sets, one with 5 products⁹ which we call HL (Hu and Liu, 2004) and one with 9 products¹⁰ which we call DL (Ding et al., 2008). The products are mostly electronics (cameras, cellphones, DVD and mp3 players), but some reviews also discuss other less related products, e.g., a garbage bin or anti-virus software.

The original data is annotated with polarity at the aspect level. To create sentence polarity annotations, we take the aspect label as sentence label if there is only one aspect or all aspect labels have the same polarity. If a “*but*” separates two aspects of conflicting polarity, the two parts of the sentence are split and separately annotated. If no splitting is possible or there is no annotated aspect, the sentence is ignored.

To be able to more directly evaluate the consistency classification part of our system, we have annotated both data sets on the level of all found sentiment words from the MPQA dictionary. The annotation was done by our graduate student of computational linguistics. The annotator had to choose one of the labels **consistent**, **inconsistent** or **non-sentiment**. Unclear cases were decided in discussion with the author.

Consider an example sentence with all sentiment words identified and numbered:

(3.15) “The reception is **good**_[0] and I had no **problems**_[1] with the handset **pressed**_[2] **against**_[3] my ear.”

The annotator chose **consistent** for word [0] “*good*”, **inconsistent** for word [1] “*problems*”, and **non-sentiment** for [2] “*pressed*” and [3] “*against*”.

Some statistics about the data can be found in Table 3.3. The second row contains the number of sentences that contain at least one sentiment word, as these are the only ones we can only compute a useful polarity score. Sentences without sentiment words are ignored for the evaluation. The numbers for found (in)consistent sentiment words refer to the words automatically extracted with the method presented in Section 3.2.3, the numbers for annotated (in)consistent sentiment words to our annotation. For the evaluation of word-level consistency classification, words annotated with **non-sentiment** are mapped to **consistent** (majority class).

Experimental setup. We evaluate on two different tasks corresponding to the two levels of annotation in our data. The task for word-level consistency classification on our word-level annotated data is to classify sentiment words in context as either **consistent** or **inconsistent**. For sentence-level polarity classification, the task is to classify sentences as either **positive** or **negative**.

⁹<http://www.cs.uic.edu/~liub/FBS/CustomReviewData.zip>

¹⁰<http://www.cs.uic.edu/~liub/FBS/Reviews-9-products.rar>

	HL		DL	
	MPQA	GI	MPQA	GI
polar sentences	1726		2100	
polar sentences with sentiment words	1446	1492	1734	1809
... positive sentences	948	947	1142	1164
... negative sentences	498	545	592	645
found sentiment words	2930	3438	3475	4423
... consistent words	2109	2337	2459	2875
... inconsistent words	821	1101	1016	1548
annotated sentiment words	2930	–	3475	–
... consistent words	1716	–	1884	–
... inconsistent words	257	–	323	–
... non-sentiment words	957	–	1268	–

Table 3.3.: Statistics of customer review data sets of 5 products (HL) and 9 products (DL) using the sentiment words from MPQA and GI.

We evaluate on each data set separately. We report accuracy (A), precision (P), recall (R), F_1 scores (F) for each class and macro F_1 scores (F_m) over the two classes. Accuracy is a less suitable performance measure for this task as the data set is skewed (about two thirds of sentences are positive and about 90% of words are consistent), so we mainly focus on macro F_1 scores.

Statistical significance is measured with the approximate randomization test (Noreen, 1989). We use 10000 iterations and measure whether the difference in macro F_1 score to negation voting with PRC-GOLD resp. classifier voting with BOC is statistically significant at $p < .05$.

Sentences have been parsed with the MATE dependency parser (Bohnet, 2010). For classification, we use the Stanford MaxEnt classifier¹¹ (Manning and Klein, 2003) with default settings. We use 10-fold cross-validation, folds are created randomly and numbers are averaged over 10 runs.

We present results for the rule-based negation voting approach and the machine learning classifier voting approach separately and compare to different simpler systems based on previous work. In both settings, we use the most simple BASELINE of all, standard voting, which assumes every word to be consistent, i.e., we set $s_{\text{cons}}(w) = 1$ for all words. Polarity classification is then done by counting words only with their prior polarity (cf. also Equation 3.1).

¹¹<https://nlp.stanford.edu/software/classifier.html>

3.5.2. Results and discussion

Results for negation voting

We first look at rule-based negation voting where we consider a word to be inconsistent when we find an odd number of negation cues in its context.

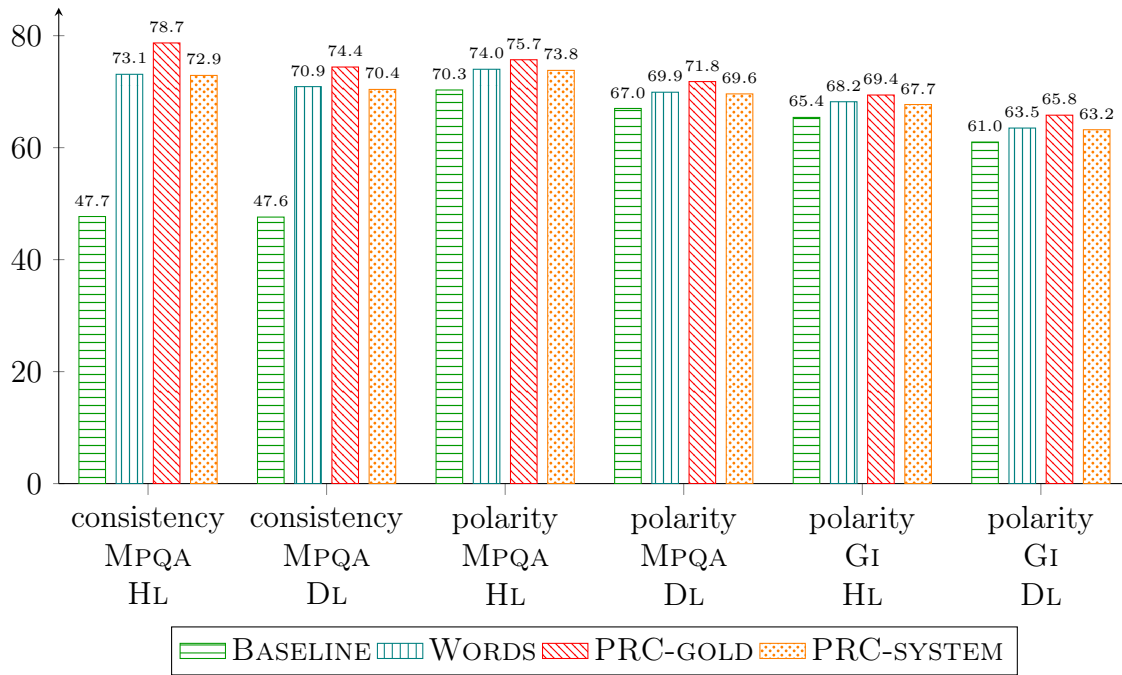
Besides the simple standard voting BASELINE, we use the common approach that uses only WORDS as negation cues. We follow previous work (Ikeda et al., 2008) and define context as the three words to the left and right of the sentiment word and use nine negation cue words: “no”, “not”, “yet”, “never”, “none”, “nobody”, “nowhere”, “nothing”, “neither”. We compare this to our system that uses either manually annotated PRCs (PRC-GOLD) or extracted PRCs (PRC-SYSTEM) as negation cues. The system checks a syntactic context of up to path-tree distance k .

Figure 3.9a contains some results for macro F_1 score on both tasks and both data sets (the complete results are in Table A.3 for consistency classification resp. Table A.5 for polarity classification). The two leftmost column blocks are on the task of word-level consistency classification with the MPQA subjectivity clues on the two data sets (we do not have word-level annotations for the GI sentiment words). The other four columns represent results for the task of sentence-level sentiment polarity classification, first using the MPQA, then the GI sentiment words.

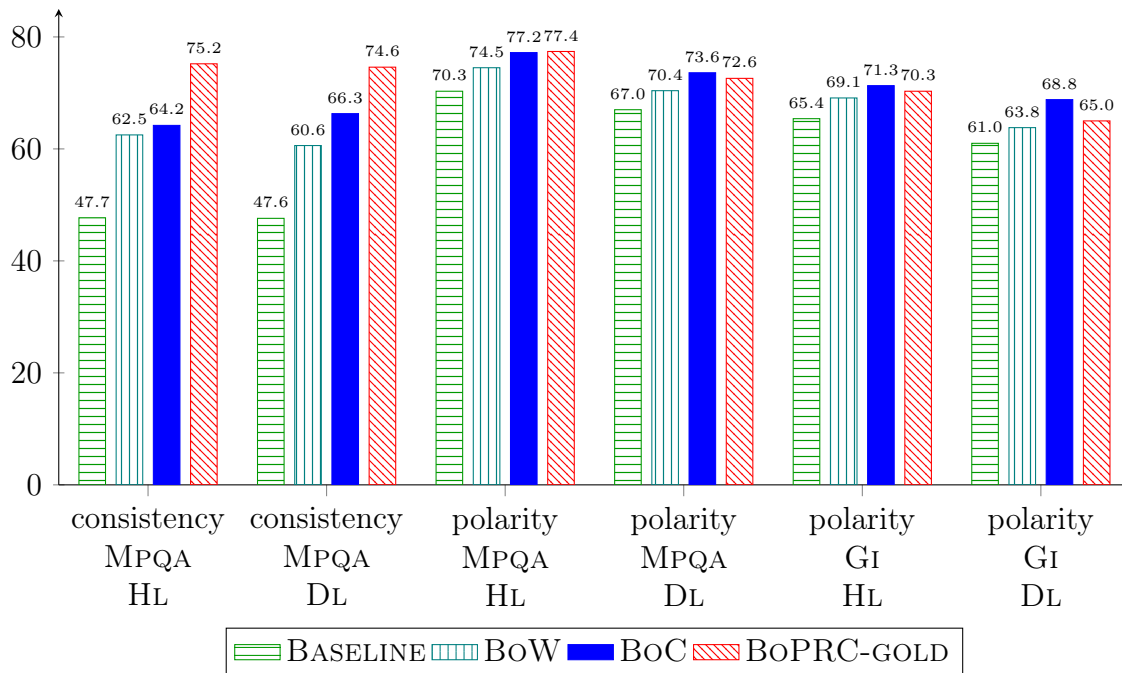
The standard voting BASELINE that classifies every word as consistent and every sentence by counting words only with their prior polarity is outperformed by all other systems. Even using only a few negation words (WORDS) improves the classification performance significantly over the baseline, both on word and on sentence level. The results are given for using three words to the left and right as context. We have varied the number of context words to use, but this setting yields the best performance, confirming the results of previous work by Ikeda et al. (2008).

When we use syntactic context and check for the presence of gold PRCs (PRC-GOLD), all results improve even further. The plot contains the result for using a path-tree distance of $k = 3$, which is the best result, but differences to using $k = 2$ or $k > 4$ are minimal. The noisy PRCs extracted with our system (PRC-SYSTEM) achieve a similar performance as WORDS (the difference is never significant). For such a noisy set (only 30% of the extracted constructions actually are PRCs!), this is a promising result. The patterns are the same for both sentiment dictionaries, but using the MPQA subjectivity clues generally gives better results.

Some examples where PRC-GOLD correctly identifies words as (in)consistent where WORDS fails to do so:



(a) Rule-based negation voting



(b) Machine learning based classifier voting

Figure 3.9.: Results results (macro F_1 score) on the customer review data sets (HL, DL) with MPQA and GI sentiment words on the tasks of word-level consistency and sentence-level polarity classification.

- (3.16) a. “1 month, no **problems**_{neg}, **great**_{pos} phone i’m very **pleased**_{pos} with my 6610 phone.” (positive)
- b. “also, the player sometimes freezes, not a very big **problem**_{neg}, but can also be **annoying**_{neg}.” (negative)
- c. “1) a spare battery would have been **great**_{pos}.” (negative)

In Sentence 3.16a, the word “problems” is correctly identified as inconsistent by both systems, but if we only look at WORDS, the word “great” is also identified as inconsistent, while our system recognizes that only the noun “problems” is in the scope of “no”. The sentence classification is not affected, it is positive in both cases. Conversely, in Sentence 3.16b, the word “problems” is not identified as inconsistent by WORDS, because the “not” is too far away. In the parse tree, it is a direct dependent of the noun and correctly identified by the construction N>not_ADJ. In Sentence 3.16c, using PRCs picks up the construction “would have been” (ADJ<V<V<would_0) as a polarity reversing construction for the sentiment word “great” in its scope.

We did some additional experiments with the PRCs extracted with f_{rel} and MI (not shown). Many of PRC extracted with MI are not actually PRCs, but words that indicate consistent context, so as expected performance drops sharply, for word-level to the baseline, for sentence-level even below (to a macro F_1 score of about 50 to 55). When we use the PRCs extracted with f_{rel} , results do not drop as sharply, but are still below using the PRCs extracted with MI+ or using WORDS. For consistency classification, F_1 score is about 63 and for sentence classification about 1 to 2.5 points above baseline.

Results for classifier voting

The second approach is machine learning based classifier voting, where a statistical classifier is used to determine $s_{\text{cons}}(w)$ and we use its classification confidence as score. Besides the standard voting BASELINE, we use a bag-of-words classifier baseline (BOW) which determines $s_{\text{cons}}(w)$ with the left and right context words of the sentiment word as features. This is a reimplement of “word-wise learning” from (Ikeda et al., 2008).

We first compare BOW to using syntactic context in the form of syntactic constructions (bag-of-constructions, BOC). This classifier uses all syntactic constructions (described in Section 3.2.2) that can be extracted from the sentiment word up to parse-tree distance k as features for the consistency classification of a given sentiment word.

As a final system, instead of using all constructions as features, we use only those that are PRCs (bag-of-PRCs, BOPRC). Our intuition for this system is that as only polarity reversal is marked, PRCs should be all that is needed to identify inconsistent words. All other constructions may occur in consistent or inconsistent context and their occurrence

does not carry any meaningful information to distinguish the two. Similar to the setup for negation voting, we compare our noisy system PRCs (BOPRC-SYSTEM) to an upper bound on performance with gold PRCs (BOPRC-GOLD).

Figure 3.9b contains results for macro F_1 score for the two tasks on both data sets in the same format as for negation voting (the complete results are in Table A.3 resp. Table A.5). We show results for a context of $k = 3$, the best performing setting.

Ikeda et al. (2008) report accuracy for sentiment-level polarity classification on the HL data with the GI sentiment words. They give an accuracy of 71.6 for polarity classification with the standard voting baseline, 73.3 for negation voting with words and 78.3 for “word-wise learning”. Our corresponding accuracies are 70.8 for the baseline, 72.9 for negation voting WORDS and 72.9 for classifier voting BOW (cf. Table A.5, the numbers shown in Figure 3.9b are F_1 scores). There are slight differences to our work in the dictionary setup and preprocessing (they report about 600 more negative words, but find 200 sentiment words less than we do), but these should not account for such a big difference in the results, so the reason for the discrepancy is unknown.

All versions of classifier voting improve upon the standard voting BASELINE that classifies every word as consistent and every sentence by counting words with their prior polarity. The BOW classifier is always outperformed by the BOC classifier, but the difference is not always significant. The reasons for the improvements are similar to those for negation voting. But because of the real-valued consistency scores, the impact of a changed label on word-level may be smaller on sentence-level than in negation voting, so the performance difference between systems is not as pronounced.

Telling the BOW classifier which features are important (BOPRC-GOLD) improves performance significantly by a large margin on word-level classification, but this improvement does not carry over to the sentence polarity classification. On sentence level, there is only a slight improvement for the HL data and MPQA sentiment words (not significant), for the other settings the results drop to somewhere in the middle between BOW and BOC. The main difference to BOC is that BOPRC-GOLD classifies less words as inconsistent, BOC assigns the label `inconsistent` to about 370 to 390 words, while BOPRC only to about 210 to 230 words. Another difference is that BOPRC-GOLD picks up some more infrequent constructions, as seen in this example:

(3.17) *“this phone won me over, and a big seller was the size; it fits **nicely**_{pos} into any pocket without **falling**_{neg} out.”* (positive)

The sentiment word “*falling*” is reversed by the construction `V<without_PR`. This construction occurs only three times in the training data, so BOC is not able to pick it up as a relevant feature for polarity reversal and “*falling*” is classified as consistent.

But the construction is contained in the gold PRC list, so BOPRC-GOLD manages to classify the word as inconsistent. On sentence level, for both systems the sentence is classified as positive with a score of 0.31 (BOC) resp. 1.26 (BOPRC-GOLD).

The results for BOPRC-SYSTEM are not shown in Figure 3.9b, as the difference to BOPRC-GOLD is always only 0.5 points F_1 score or less. Still, for such a noisy set, this is actually surprisingly good. We have also again repeated the experiments with the PRCs extracted with MI and f_{rel} . MI results in a sentence-level performance similar to MI+, occasionally even better, while f_{rel} gives results comparable to the BOW classifier.

In summary, we see improvements over the baseline with all types of context information. In rule-based negation voting, the best results overall are achieved by using gold PRCs. Using automatically extracted PRCs is most of the time worse than using a very small number of keywords. For machine learning based classifier voting, on word-level consistency classification using PRCs whether gold or automatically extracted beats every other method by a large margin. This does not carry over to sentence-level polarity classification though, where the best result is achieved by using all constructions (BOC).

3.6. Summary

This chapter addressed negation or polarity reversal which is our first example of a complex verbalization of opinions. We presented a supervised machine learning based approach to detect whether a sentiment word is consistent or inconsistent with its prior dictionary polarity in a specific sentence context. As our first contribution, we have shown that the use of paths through the dependency parse as features for the classifier can improve performance on word-level consistency classification and on sentence-level polarity classification. As a second contribution, we presented first steps towards automatically extracting polarity reversing constructions (PRCs) from sentences annotated with polarity. Even though the set of extracted PRCs is noisy, we can successfully use them as features for our classifier to improve classification performance.

4. A corpus of comparisons in product reviews

The content of this chapter has been published in (Kessler and Kuhn, 2014a).

4.1. Introduction

We have addressed the topic of negation as our first example for a complex verbalization in the last chapter, Chapter 3. Starting from this chapter, the remainder of this thesis focuses on comparisons, our second example of a complex verbalization. The analysis of comparisons is of interest because they are relatively frequent in reviews (5-10% of sentences in our data). Also, comparisons are presumably the most useful kind of expression when it comes to supporting a process of choice, as they provide explicit comparative assessments which can provide arguments to enable a user to make a decision between several products. This chapter presents a corpus of manually annotated comparisons that we have created as a basis for our work.

Sentiment analysis has attracted a lot of research in recent years and there is a wealth of resources that are annotated with sentiment information on different levels of detail, but comparisons have not received a lot of attention (see Section 2.5 for an overview). Consequently, while there is a large number of annotated resources available for sentiment analysis, most annotations are based on the standard approach for aspect-based sentiment analysis that presumes one target for each sentiment expression. This type of annotation is insufficient to model the information contained in comparisons. To our knowledge, to date there are only two popular English sentiment corpora in the domain of reviews which include detailed annotations for comparisons and are publicly available (J. Kessler et al., 2010; Jindal and Liu, 2006b). In these existing corpora many decisions were left to the annotator, which leads to considerable variety in the data. Additionally, each data set is a relatively small source of training examples for machine learning and combining the two leads to a high degree of heterogeneity, since not only the domains but also the annotations schemes vary.

To overcome these limitations, we have decided to create our own data set of annotated comparisons. We have chosen to annotate sentences from English reviews about digital cameras taken from the data provided by Branavan et al. (2009). A comparison consists of several parts that must be annotated in order to get meaningful information. Consider the following examples:

- (4.1) a. “[It]_{E1} had a **better** [lens]_A than [the D60]_{E2}.”
 b. “[It]_{E1} had a **more** [durable]_S [lens]_A than [the D60]_{E2}.”

The main anchor for a comparison is the comparative predicate, the word or phrase that expresses the comparison. In the example, the predicate “*better*” resp. “*more*” is marked in bold. A comparison involves two entities that are set into a relation, “*It*” and “*the D60*” in the examples, marked in brackets with a subscript E1 and E2. In our data, most of the entities are products, specifically digital cameras. Two entities are not usually compared in their entirety, but in some aspect, the component “*lens*” in the example, marked with a subscript A. In the special case of multiword predicates like “*more durable*”, a fourth argument type named scale is added to contain the modified adjective or adverb, “*durable*” in the second example, marked with a subscript S.

A sentence may contain none, one or several comparisons. For our purposes we define a comparison to be any statement about the similarity or difference of two entities (Jindal and Liu, 2006b). Comparative sentences in the linguistic sense (“*X is better than Y*” or “*X is the best*”) are included in this definition and indeed many comparisons are of this form. But comparisons in user generated texts also contain many more diverse statements that we also include in our definition, e.g., “*X blows away all others*”, “*X and Y have the same B*”, “*X wins over Y*”.

In the following, we first give a short overview of the existing data sets and their annotations. Next, we present the process and the annotation scheme used to create our corpus of comparisons in camera reviews. We then report the results of an agreement study, give some statistics about our data as compared to the other existing data and finally discuss some open issues and questions. To our knowledge, our corpus is the largest source of comparison sentences in reviews to date. The data and the annotation guidelines are publicly available on our website¹.

4.2. Existing data sets

To our knowledge, to date there are two popular sentiment corpora in the domain of reviews which include detailed annotations for comparisons and are publicly available.

¹<http://hdl.handle.net/11022/1007-0000-0000-8E72-0>

Both corpora identify a comparative predicate as the anchor of a comparison, and for each predicate two entities and an aspect, but they differ in many details. Also, some decisions were left to the annotator, which leads to considerable variety in the data.

Jindal and Liu data set (J&L). Jindal and Liu (2006b) created the first data set specifically for the task of identifying comparisons². The data set contains approximately 650 comparison sentences from reviews, blog posts and forum discussions about various topics ranging from digital cameras over processors to soccer and soft drinks.

In the J&L data, a comparison is represented as a tuple (relationWord, features, entityS1, entityS2, type). Each part of the tuple can be empty or occur multiple times. A sentence can contain several comparisons. There are no restrictions on the part of speech tags or possible comparison expressions. Arguments are only annotated inside the sentence. The *relation word* is the “keyword used to express a comparative relation in a sentence”, i.e., corresponding to what we call the comparative predicate. The part *features* corresponds to what later publications call the aspect. Entities in the J&L corpus are annotated as *entity S1* or *S2* based on the order of appearance in the sentence, the preferred entity is not marked. Every comparison is assigned one out of four types of comparisons (1=equative, 2=non-equal gradable, 3=superlative and 4=non-gradable, see also Section 4.4). Predicates are mostly single tokens, but there are some multiword predicates annotated in the data, e.g., “*as good as*” or “*one of the few*”. One comparison may have several predicates.

The following shows some example sentences from the data and their annotations (sentences are provided in tokenized form in the original data):

- (4.2) a. “*it ’s because the [256 player]_{entityS1} uses [flash memory]_{features} (**smaller** / **lighter**) and the [30gb]_{entityS2} uses a hard drive .*” (non-equal gradable)
- b. “*the [flash card memory]_{entityS1} is **more** [expensive]_{features} but is very light and compact .*” (non-equal gradable)
- c. “*to my ears , a [160 kbps wma file]_{entityS1} [sounds]_{features} just **as good as** the [source cd]_{entityS2} , ...*” (equative)
- d. “*[this]_{entityS1} phone is **one of the few** phones that have an [fm radio tuner]_{features} built in .*” (superlative)

JDPA corpus (J-A, J-C). The JDPA (J.D. Power and Associates Sentiment Corpus) corpus³ by J. Kessler et al. (2010) consists of blog posts about cameras (about 500

²<http://www.cs.uic.edu/~liub/FBS/data.tar.gz>

³<http://verbs.colorado.edu/jdpacorpust/>

comparison sentences, abbreviated J-C in the following) and cars (about 1100 comparison sentences, abbreviated J-A in the following). The corpus contains detailed information about entities and sentiment expressions. Comparisons are annotated (annotation class “Comparison”), but were not the focus of the annotation.

The anchor of the comparison is a comparison word, corresponding to our comparative predicate. According to the annotation guidelines, only comparative adjectives and adverbs (called “quantifiers” in the guidelines) are regarded as comparisons. In practice, many more diverse expressions are annotated, especially in the later batches of the data. A sentence can contain several comparisons. Comparison annotations have five slots corresponding to the two entities (*Less*, *More*), the aspect (*Dimension*) and a basic indicator for the comparison type (*Same*). Not all slots have to be filled, but each slot can have only one value. Arguments may be outside the current sentence, but need to be in the same review. Entities are annotated explicitly as the preferred (*More*) or non-preferred entity (*Less*). The *Same* slot is a Boolean value that is set to true if “two Entity Mentions are equal in the trait in question”, i.e., there is no ranking introduced. The assignment of entities to the slots for the preferred and non-preferred entity has to be made even if the comparison does not introduce an ordering, the order of assignment into the two slots is not defined by the guidelines in this case. In some cases, the comparison word is also marked as the aspect, e.g., in the phrase “*higher seating positions*” the word “*higher*” should be annotated as both predicate and aspect. The phrase “*seating positions*” is the target of the sentiment expression annotated at the same point. Predicates can be multiword, but are mostly split into predicate and aspect. A phrase like “*more modest looking station wagon*” would have “*more*” as the predicate and “*modest*” as the aspect, the phrase “*as good as*” would have the first “*as*” as the predicate and “*good*” would be the aspect.

The following shows some example sentences from the J-C camera data and their original annotations (J-A sentences and annotations are similar):

- (4.3) a. “[350D]_{More} is actually even [**smaller**]_{Dim} then [300D]_{Less}.” (same=false)
 b. “[It]_{More}’s **more** [expensive]_{Dim} than the budget-priced [L11]_{Less}, but it’s still one of the most affordable cameras available with manual exposure controls.” (same=false)
 c. “Often, [they]_{Less} are **as** [cheap]_{Dim} as [anything]_{More} found online.” (same=true)

4.3. Data sources and annotation procedure

For our own annotated corpus, we have chosen to annotate camera reviews in order to get data from the same domain as the existing corpora. In this way the JDPA camera

data set and the camera part of the J&L data could be added to our new data set to form a larger (albeit noisier) training set for machine learning.

As most sentences in a typical review do not contain a comparison, we use a two-stage annotation process. First, we use crowdsourcing to identify sentences that with high probability contain a comparison. Only these sentences were then passed on to the second stage and annotated in more detail by a trained annotator.

Data sources and preprocessing. We base our corpus of comparison sentences on the camera data set provided in XML format by Branavan et al. (2009)⁴. They downloaded a set of camera reviews from `epinions.com` and separated the review text from the other information. Reviews that were duplicates, off-topic or not English were manually removed when the annotators detected them (48 reviews in total). We removed HTML tags from the review texts and used Stanford CoreNLP⁵ (Manning et al., 2014) to automatically do sentence segmentation and tokenization.

The first part of the data which we have published at LREC 2014 covers the reviews from the top of the file until the review with id 649 (the listing of reviews is not always according to the id). The second part covers part from the next review with id 694 until the review with id 959. Not counting the ignored reviews, in total we have processed 930 reviews with over 17.000 sentences.

Identifying sentences that contain a comparison. We decided to use crowdsourcing for the task of deciding whether a sentence contains a comparison. We designed a HIT (Human Intelligence Task) and uploaded it on Amazon Mechanical Turk (AMT)⁶. The workers were given short instructions about their task and a few examples of comparisons and non-comparisons. Every sentence was annotated by two AMT workers. Possible labels were “comparison”, “no comparison” or “not sure”. To discourage the use of “not sure”, workers were asked to provide feedback on why they chose this value. Figure 4.1 shows a screenshot of the interface for the annotation task.

A batch of 500 sentences was annotated on AMT in about one hour. The results imply that the task is difficult for AMT workers. The workers agreed on “no comparison” for about 40% of sentences and on “comparison” for about 20% of sentences. Compared to the 5–10% of comparison sentences we would expect to find, this corresponds to high recall. If both AMT workers chose “comparison”, the sentence was passed on to the

⁴<http://groups.csail.mit.edu/rbg/code/precis/>

⁵<http://nlp.stanford.edu/software/corenlp.shtml>

⁶<http://www.mturk.com>

Find comparison sentences

Your task is to decide for every one of the 5 sentences given below if the sentence contains at least one comparison.

Examples for sentences that **contain** a comparison:

- Slower than a lot of other lenses, because it has more glass.
- This is the best camera ever!
- BTW, the A100 uses the same battery as the R1.
- The biggest difference is the quality of the jpegs.
- Canon's IS and Nikon's VR cant match this.

Examples for sentences that **do not contain** a comparison:

- This is my first digital camera, and i am very pleased with it.
- I am more than willing to help you.
- The finish is more mirror than silver.
- I bought this camera as a replacement for the sd700.

You must select one value for each sentence. Select

- "Comparison" if the sentence contains a comparison, it compares the product with something else even if the other object is not explicitly mentioned.
- "No comparison" if there is no comparison.
- "Not sure" if the sentence is not complete, incomprehensible or you find it very hard to decide (you can write the reason into the feedback field below). You are not allowed to chose this value for every sentence.

Does this sentence contain a comparison?

1. [Using the Sony name will probably give it an `` entry level " status .](#)
 Comparison No comparison Not sure
2. [It 's a great toy for people who will buy their first DSLR .](#)
 Comparison No comparison Not sure
3. [But dont expect the camera to attract Canon or Nikon enthusiasts .](#)
 Comparison No comparison Not sure
4. [Canon 's IS and Nikon 's VR cant match this .](#)
 Comparison No comparison Not sure
5. [I definately recommend this camera .](#)
 Comparison No comparison Not sure

Provide feedback (if you want):

Submit

Figure 4.1.: Screenshot of the AMT annotation interface for the first stage of annotation that finds sentences containing comparisons (taken on February 2nd, 2014).

second stage, if both chose “no comparison” the sentence was discarded. In the rest of the cases (about 40% of sentences), the AMT workers did not agree or both workers chose “unsure”. These sentences were roughly checked for obvious non-comparisons and then passed on to the second stage.

Fine-grained comparison annotation. The more fine-grained annotation was carried out by three annotators hired and trained specifically for the task. One was a graduate student of computational linguistics, one an undergraduate student of media and the last one a graduate student of computer science. All had an advanced level of English, but none was a native speaker.

The annotators were given annotation guidelines with detailed instructions. We chose 30 sentences from our data as a training set. This set contained all types of comparisons and was annotated by the author as well. Each annotator had to first annotate this training set and any disagreements with the author were discussed.

After training, the actual annotation was carried out by the annotators independently. The annotators only saw the sentence they were currently annotating, without the context of the review. Annotators had the possibility to decide that the sentence is not a comparative sentence. This is necessary as the first stage was designed to give high recall, not precision. Figure 4.2 shows the annotation of a comparison sentence with our tool. Apart from the annotations, annotators were encouraged to provide additional feedback in hard cases. These cases were discussed with the author and the feedback was used to refine the annotation guidelines.

4.4. Annotation scheme

The data to be annotated consists of sentences from user generated content, namely reviews of digital cameras. The goal of the annotation is to provide fine-grained information on comparisons. We did not annotate any other information besides comparisons.

Recall that we define a comparison as any statement about the similarity or difference of two entities. A sentence may contain none, one or several comparisons.

The main anchor for a comparison is the comparative predicate. It has the following arguments (relations): The two entities that are being compared (E1, E2), and the aspect they are compared in (A). In some special cases a fourth argument scale for modified adjectives/adverbs (S) is added. Predicates and arguments are annotated as token spans in the same sentence, we do not annotate parts of tokens or cross-sentence relations. In addition to the arguments, for each predicate we annotate the type of the comparison

```

==== Sentence to be labeled (nr. 74) ====

1_The 2_Canon 3_EOS 4_40D 5_Digital 6_SLR 7_ultimately 8_delivers 9_better
10_photo 11_quality 12_than 13_it 14_s 15_competitors 16_.

Please identify the comparative predicate in the sentence. Write the word as it
occurs in the sentence (with ID). If there is no further predicate, leave empty.
Use 'q' to quit.
Comparative predicate: 9_better

Please label the scale part (if existing). Write the word as it occurs in
the sentence (with ID). Use 'q' to quit.
Scale:

Please identify the arguments of the predicate you just labeled in the sentence.
Write the word(s) as it occurs in the sentence. If the argument does not occur,
leave empty. Separate several arguments with ' | '.
Please also label the entity type. Use 'p' for product(s), 'g' for a set/group
of products, 'f' for feature/aspect(s), 'c' for company, 's' for some standard,
'o' for other (specify in notes). Use 'q' to quit.
Entity 1: 1_The 2_Canon 3_EOS 4_40D 5_Digital 6_SLR
Type Entity 1: p
Entity 2: 13_it 14_s 15_competitors
Type Entity 2: g
Aspect: 10_photo 11_quality

Please label the relationship of the entities to one another. Use 'r' if one
entity is ranked as better than the other, 's' if one entity is set above all
others, 'e' if both entities are rated as equal, 'd' if a non-graded difference
is expressed. Use 'q' to quit.
Comparative Type: r

Please label which entity is evaluated as better. Use '1' if E1 is rated better
than E2, '2' if E2 is rated better than E1, 'x' else. Use 'q' to quit.
Fine-grained Type: 1

1_The 2_Canon 3_EOS 4_40D 5_Digital 6_SLR 7_ultimately 8_delivers 9_better
10_photo 11_quality 12_than 13_it 14_s 15_competitors 16_.

Please identify the comparative predicate in the sentence. Write the word as it
occurs in the sentence (with ID). If there is no further predicate, leave empty.
Use 'q' to quit.
Comparative predicate:

Please tell me if there is a problem with the sentence (splitting, tokenization,
incomplete, ...). If there is none, leave empty. Use 'q' to quit.
Annotation note:

```

Figure 4.2.: Example for the fine-grained annotation of comparisons in a sentence with our command-line based annotation tool.

and the types of the involved entities. The parts of comparison to be annotated are discussed in more detail in the following subsections.

Comparative predicate. The central part of any comparison is the *comparative predicate*. The comparative predicate is the syntactic marker that introduces a comparison. As an illustration, consider the following example sentences from our data (all examples are presented with original spelling and punctuation):

- (4.4) a. “But [this new XT]_{E1} compared to [the old rebel]_{E2} has MUCH **better** [picture quality]_A.”
 b. “[The XT]_{E1} **beat** [the 300D]_{E2} in the [file writing]_A department as well . . .”
 c. “The biggest **difference** is the [quality of the jpegs]_A.”
 d. “By the way, [Nikon]_{E1} is at the **top of the line** in [flashes]_A.”
 e. “[It]_{E1} shared the **same** [sensor]_A as [Nikon D200]_{E2}, and [the latest D80]_{E2}.”
 f. “In fact I think [the D80]_{E1} is **better** at [handling noise]_A and [suppressing banding artifacts]_A at higher ISO’s.”
 g. “I couldn’t be happier!”

Predicates can be of any part of speech, e.g., adjectives (Sentence 4.4a), verbs (Sentence 4.4b), nouns (Sentence 4.4c). A sentence may contain more than one comparative predicate. We allow annotators to annotate multiword expressions. This mainly concerns expressions such as “*top of the line*” in Sentence 4.4d.

Comparisons can express some personal opinion or belief (subjective, Sentence 4.4a), or state verifiable facts (objective, Sentence 4.4e). We annotate both subjective and objective comparisons. We do not include expressions that on the surface look like comparisons, but are used as descriptions of environments or states, i.e., “*at higher ISOs*” in Sentence 4.4f or “*happier*” in Sentence 4.4g.

Scale (S). There is a limited number of predicates that are function words and do not by themselves contain the information about what distinguishes the entities. Consider these sentences from our data:

- (4.5) a. “Only [the S3]_{E1} has a potentially **better** [movie record mode]_A.”
 b. “. . . [the SD800]_{E1} has a **more** [powerful]_S and overall flexible [movie capture mode]_A.”
 c. “That’s a pretty [good]_S [price]_A **compared** to [everything else that I’ve seen]_{E2} . . .”
 d. “[it]_{E1}’s just as **as** [capable]_S as [the D200]_{E2}.”

The annotation for the predicate “*better*” in Sentence 4.5a is straightforward and does not require an additional argument. We would like to have the annotations for the comparison “*more powerful*” in Sentence 4.5b parallel the annotations for the comparison

“better”. One possible way to go would be to annotate “more powerful” as a multiword predicate. A second possible way is to split the predicate into two parts, annotate the function word “more” as the predicate and the modified adjective “powerful” as an argument which we call scale (S). We chose to split the predicate as this allows us to also capture cases where the two parts are not adjacent (Sentence 4.5c). There is a limited number of predicates that allow the annotation of a scale argument. Besides ranked comparisons with “less” and “more”, another frequent predicate is “as” when used to introduce an equative comparison (Sentence 4.5d).

Entities (E1, E2). A comparison involves two *entities* that are compared with each other. Consider the following example sentences:

- (4.6) a. “[350D]_{E1} is actually even **smaller** than [300D]_{E2}”
 b. “By the way, [Nikon]_{E1} is at the **top of the line** in [flashes]_A.”
 c. “**Best** [battery life]_A.”
 d. “[It]_{E1} shared the **same** [sensor]_A as [Nikon D200]_{E2}, and [the latest D80]_{E2}.”
 e. “[[All three models]_{E1}]_{E2} are extremely **close** in terms of [price]_A and [features]_A.”

Most often the entities in our data are products, e.g., the two cameras “350D” and “300D” in Sentence 4.6a. We decided to annotate entities based on the order of appearance in the sentence. The first entity is annotated as entity 1, the second as entity 2. The information which entity is preferred if the comparison introduces a ranking between the entities is included in the comparison type. This style of annotation is much easier for annotators, as they are not forced to choose a “better” and “worse” entity if there is no obvious ranking introduced by the expression.

One or both of the entities may be implicit (entity 2 in Sentence 4.6b, both entities in Sentence 4.6c). An entity can consist of a group of products that are listed individually. As an example consider Sentence 4.6d, where entity 1 “it” (presumably the camera under review) is compared to two other cameras, “Nikon D200”, and “the latest D80”. Both cameras together make up entity 2, and each listed item is annotated individually.

If all compared entities are ranked as being on the same level, they are sometimes referenced together, e.g., using “all” or “both”. In this case, the plural reference is annotated as both entity 1 and entity 2 (Sentence 4.6e).

Aspects (A). In most sentences one attribute or part of a product is being compared. We follow the terminology of aspect-oriented sentiment analysis and call this the *aspect*⁷ (cf. Section 2.2.5). The notion of an aspect includes everything relevant to the product,

⁷Other terms are “feature”, “attribute” or “dimension”.

i.e., parts, properties or attributes of the product, results of using the product, and actions performed with it. Consider the following examples:

- (4.7) a. “**Best** [battery life]_A.”
 b. “[The G9]_{E1} is much **easier** to [shoot]_A with.”
 c. “[350D]_{E1} is actually even **smaller** than [300D]._{E2}”
 d. “Many digital SLRs have [sensors]_{E1} whose [size]_A is **smaller** than [that of a 35mm film frame]_{E2}.”
 e. “In fact I think [the D80]_{E1} is **better** at [handling noise]_A and [suppressing banding artifacts]_A at higher ISO’s.”

Like with entities, there can be more than one aspect compared at the same time, e.g., “handling noise” and “suppressing banding artifacts” in Sentence 4.7e. Aspects may not always be explicit, e.g., “small” implies the aspect size. We only annotate explicit aspects (Sentence 4.7c vs. Sentence 4.7d).

Entity type. For each entity in a comparison we annotate the entity type. Most of the entities are products (cameras in our case), but we distinguish whether an entity is a single product (E1 in Sentence 4.8a) or a set of products (E2 in Sentence 4.8a). Entities can also be a reference to a company (E1 in Sentence 4.8b) or a reference to a general standard (E2 in Sentence 4.8b).

- (4.8) a. “[It]_{E1} shared the **same** [sensor]_A as [Nikon D200]_{E2}, and [the latest D80]_{E2}.”
 b. “By the way, [Nikon]_{E1} is at the **top of the line** in [flashes]_A.”

Entities can also be of the type “aspect”. While this may sound confusing at first, this occurs because aspects form a hierarchy. It is possible to talk about an aspect of an aspect of the product. Consider the following sentences:

- (4.9) a. “Performance: [The D80]_{E1-product} uses essentially the **same** [sensor]_A as [the D200]_{E2-product}.”
 b. “[This full-sized framed sensor]_{E1-aspect} contains the exact **same** [megapixel density]_A as [the one in the Rebel XT and the 20D]_{E2-aspect} . . .”

In Sentence 4.9a, “sensor” is the aspect that is compared in a comparison between two cameras. In Sentence 4.9b, two sensors are compared in their aspect “megapixel density”. The two entities should get the type “aspect” in this sentence. The distinction between entity and aspect for the purpose of our annotation is not only dependent on semantic class (e.g., camera or camera part), but also involves the function in the sentence. Distinguishing between the two may be very relevant for a system that lists all differences between two products A and B. For sentences like Sentence 4.9b we would

want to list the comparison of an aspect of A under the product A itself. Whenever an entity has the type “aspect”, it would serve as an indicator that some linking of the aspect back to the product it belongs to has to be done.

In total, we distinguish six possible types for entities: product, set of products, standard, company, aspect or other. This list of types is tailored to our domain of product reviews and will need adaptation for other domains.

Comparison type. As discussed in more detail in Section 2.5.3, comparisons have two main types, gradable and non-gradable (Jindal and Liu, 2006b; Liu, 2015). A gradable comparison expresses an ordering relationship of the entities being compared. This ordering relationship can have three forms, it can set one entity over the other (non-equal gradable or **ranked** comparison, Sentence 4.10a), one entity above/below all others (**superlative** comparison, Sentence 4.10b), or declare all entities as being equal (**equative** comparison, Sentence 4.10c). The first two relations also have two subtypes that clarify the direction of the relation (E1 is superior/inferior to E2), i.e., the order of preference for the entities.

Non-gradable comparisons express a difference between two entities, but do not rank the entities. We annotate non-gradable comparisons only if there is a direct comparison between two entities in an aspect they share (aspect difference, Sentence 4.10d). We do not annotate statements that list existing or missing aspects, even if two entities occur in the sentence, e.g. “X has A, but Y not” or “X has A, but Y has B”.

As a result, like in the J&L corpus, we distinguish four types of comparisons: ranked, superlative, equative comparisons, and non-graded differences. In addition to the J&L annotation of comparison types, we annotate the direction (superior/inferior) for ranked and superlative comparisons.

- (4.10) a. “In fact I think [the D80]_{E1} is **better** at [handling noise]_A and [suppressing banding artifacts]_A at higher ISO’s.” (**ranked**, E1 superior to E2)
- b. “**Best** [battery life]_A.” (**superlative**, E1 superior to E2)
- c. “[It]_{E1} comes with the **same** [stabilizer technology]_A that [the Nikon D200]_{E2} has.” (**equative**)
- d. “The biggest **difference** is the [quality of the jpegs]_A.” (aspect **difference**)
- e. “The ‘SX1’ is basically an SX10, but adds a CMOS sensor instead of CCD, and a full 1080 HD video mode.” (existence **difference**)

The type of comparison cannot be determined solely on the basis of the predicate. Syntactic context, especially negation, changes the type of comparison. As an example

take Sentence 4.5d which is an equative comparison. If we use “*not as capable*” instead of “*as capable*”, the result would be a ranked comparison where entity 2 is preferred.

Sometimes the direction of a ranked comparison is unclear, especially with predicates where the direction depends on the aspect, like “*smaller*” or “*higher*”. In such cases, the annotators are asked to rely on their world knowledge (e.g., high resolution is good, high price is bad) or any context available in the sentence.

4.5. Analysis of the data

4.5.1. Inter-annotator agreement

All annotators had to annotate a set of 100 sentences for the purpose of calculating agreement between annotators as a measure of consistency. These sentences do not include the annotator training set. All of the 100 sentences were judged by the AMT workers to contain a comparison in the first annotation stage.

For the annotation of text spans where each annotator individually picks some words from the sentence instead of assigning a label from a predefined set, we follow J. Kessler et al. (2010) and use text span agreement. For sets of annotations X and Y by annotators x and y , the agreement of x to y is calculated as

$$\text{agr}(x||y) = \frac{|X \text{ matches } Y|}{|X|} \quad (4.1)$$

We consider two varieties of matching. In the strict version agr_s , two spans are considered to match only if they are exactly the same. In the lenient version agr_l , two spans are considered to match if they have at least one overlapping token. Only matches of non-empty text spans are counted. Text span agreement is calculated for each pair of annotators, we report the average over all pairs of annotators. We only compare spans of the same type, if one annotator annotates some span as an entity and another annotates the same span as an aspect, this is not a match.

For categorical label assignments we measure observed agreement A and chance-corrected agreement with Cohen’s κ (Cohen, 1960) for each pair of annotators as

$$\kappa = \frac{P(A) - P(E)}{1 - P(E)} \quad (4.2)$$

We report the averages of pairwise A and κ . All metrics are calculated only on items that have been annotated by both annotators, i.e., for the comparison type the

Categorical:	A	κ	# agreed
Sentence level	0.8800	0.4448	82
# comparisons	0.8327	0.6314	58
Comparison type	0.7872	0.6065	57
Type entity 1	0.8442	0.6512	45
Type entity 2	0.8102	0.6478	29
Text spans:	agr_s	agr_l	# agreed
Predicate	0.7987	0.8303	80/84
Scale	0.8943	0.8943	14/14
Entity 1	0.7551	0.8555	48/58
Entity 2	0.7280	0.8254	33/40
Aspect	0.5587	0.7547	22/31

Table 4.1.: Inter-annotator agreement on the set of 100 sentences annotated by all three annotators. All agreement numbers are averages over pairwise measures. We report observed agreement A , Cohen’s κ , strict text span agreement agr_s and lenient text span agreement agr_l . # agreed gives the number of items on which all annotators agreed (lenient match/strict match).

predicates, for entity types the entity spans have to match (lenient match). For the number of comparisons in a sentence, we put all values over 4 into the same bucket, as this is very rare.

Table 4.1 shows the results of the agreement study. The last column shows the number of items on which all annotators agreed (lenient match/strict match).

Overall, agreement is in the range of values reported for other sentiment annotation tasks. To our knowledge, there are no values reported for the annotation of comparisons in previous work. J. Kessler et al. (2010) report text span agreement for some annotations, but not for comparisons. For the task of identifying sentiment text spans which might be comparable to identifying predicates, they report a lenient agreement of 0.75. For the identification of the sentiment-target relation which may roughly be comparable to identifying the entities in a comparison, they report 0.66. The values are not directly comparable across corpora and annotation schemes, but may give a general idea about the range of expected agreement values.

On sentence-level, the task is to decide whether a sentence contains at least one comparison or none at all. In our data, agreement on sentence-level is close to 90%, but κ is relatively low at 0.45. A value between 0.4 and 0.6 is considered moderate agreement (Landis and Koch, 1977). One reason for the low κ is the very skewed distribution, as all of the sentences were selected as containing comparisons in the first annotation stage.

Of the 82 sentences that all annotators agreed on the label, 77 had the label “contains a comparison”. But the rather low agreement also confirms the results we got from the AMT experiments, namely that the decision whether a sentence contains a comparison is sometimes not as easy as it may seem. Further discussion of this point can be found at the end of this chapter in Section 4.5.3.

For the assignment of comparison type, we have an agreement value of 79% and $\kappa = 0.61$, κ values between 0.6 and 0.8 are considered substantial agreement (Landis and Koch, 1977). The main source of disagreements about the type of comparison in our data is that one annotator tended to annotate comparisons as type difference if they were ranked but the direction was not clear to the annotator.

Once it is established that there is a comparison, text span agreements are high. Disagreements mainly come from determining the exact boundary, e.g., one annotator annotated “*RAW photo quality*” as the aspect, another “*photo quality*”. Entities are easier to identify than aspects, probably because aspects require more domain knowledge.

For the assignment of entity types we again have substantial agreement. Disagreements about the entity type come mainly from two sources. The main source of disagreements is missing domain knowledge or missing context. Often entities are referred to by a model number only. When an annotator was lacking domain knowledge, such occurrences were mislabeled as ‘other’. It is also common to refer to a product by the company name only, leading to erroneous ‘company’ labels. Most of these errors can be spotted by comparing the types assigned to the two compared entities, as they should usually have the same type. Missing context or domain knowledge is responsible for almost all the cases where entities 1 and 2 have different types. In a few cases the annotation of different types reflects an error on the part of the sentence author, e.g., in Sentence 4.11a the correct reference would actually be “*the one on the 600*”.

The second source of disagreements is when entity spans match leniently, but this changes the type of the entity. Consider the annotations for the following sentence:

- (4.11) a. “... [*the 630*]_{E1}’s [*lcd*]_A seemed **less** [*sharp*]_S compared to [*the 600*]_{E2}.” (ann. 1)
 b. “... [*the 630’s lcd*]_{E1} seemed **less** [*sharp*]_S compared to [*the 600*]_{E2}.” (ann. 2)

The text spans of entity 1 overlap, so entity types are compared. But “*the 630*” refers to a product and “*the 630’s lcd*” to an aspect, so the types do not match. This example also illustrates that the differentiation between entity and aspect is sometimes not that clear-cut (further discussed in Section 4.5.3).

We have used the results from this agreement study to refine the annotation guidelines. Most importantly, we have explicitly excluded some categories of sentences that are not

comparisons, and added the possibility to leave the direction of ranked comparisons unspecified. Annotators have been asked to review their annotations for these errors.

4.5.2. Statistics and comparison to other data sets

Statistics about the IMS data. Statistics about our complete corpus can be found in Table 4.2 in the column marked “IMS”. These numbers do not include the annotator training set or the agreement set. In total we collected nearly 2200 sentences that contain at least one comparison. This is about 13% of all sentences extracted from the reviews.

The average number of comparisons per sentence is 1.27. The overwhelming majority of sentences (just over 80%) contains only one comparison, nearly all of the remainder contain 2 or 3 comparisons. The maximum number of comparisons is 17 (!), this is a sentence that consists of an enumeration of basically every aspect of a camera⁸. If we exclude this sentence, there is one other sentence with 7 comparisons and then several sentences with 4 or 5 comparisons.

There are many possible ways of expressing comparisons, therefore we observe a large heterogeneity of constructions in our data. This particularly affects verb and noun predicates, where many colloquial expressions are used (*“hammers”*, *“pwns”*, *“go head to head with”*, *“put X to the sword”*, ...). Only about 45% of comparisons have an adjective or adverb in the comparative or superlative form as a predicate. The most frequent predicates in ranked comparisons are *“better”*, *“more”*, *“as”* (in sentences with *“X is not as A as Y”*) and *“less”*. The most frequent predicates for equative comparisons are *“same”*, *“as”*, *“similar”* and *“like”*. Statements of difference often have *“difference”*, *“compared”* and *“different”* as predicate. Nearly all superlative comparisons contain an adjective in superlative form, the rest are statements like *“nothing beats X”*.

The overwhelming majority of comparisons is ranked (67%). There are far more ranked and superlative comparisons with direction superior than inferior (80% to 20% for those where a direction is given). This is consistent with the bias on positive statements repeatedly reported in sentiment analysis. Most of the comparisons have one entity 1 and one entity 2, but is also common to drop entity 2, especially for superlatives. Usually at least one aspect is present in a comparison as well, except for superlatives and differences where more than half of the instances do not contain any aspect. About 70% of the compared entities are products (one product or a set of products). The next largest type (15%) is the type “aspect”. Entities with type “aspect” mainly includes parts

⁸An excerpt: “... is **easier** to [navigate]_A, has **more** [features]_A, **more** [auto-focus points]_A (9 vs 7), **better** [quality]_A and **faster** [auto-focus]_A, ...”

	IMS	J-A	J-C	J&L
documents	978	425	180	–
all sentences	17074	13965	4972	7981
sentences containing comparisons	2193	1094	505	651
% of all sentences	12.8	7.8	10.2	8.1
sentences with 1 comparison	1762	910	406	607
... with 2 comparisons	333	146	75	41
... with 3 comparisons	67	30	14	3
... with 4 comparisons	20	5	6	0
... with ≥ 5 comparisons	11	3	4	0
% of sentences with 1 comparison	80.3	83.2	80.4	93.2
... with 2 comparisons	15.2	13.3	14.9	6.3
... with 3 comparisons	3.1	2.7	2.8	0.5
... with 4 comparisons	0.9	0.5	1.2	0
... with ≥ 5 comparisons	0.5	0.3	0.8	0
comparisons with type ranked	1861	849	453	367
... superlative	204	–	–	169
... equative	457	478	189	162
... difference	256	–	–	–
% of comparisons with type ranked	67.0	64.0	70.6	52.6
... superlative	7.3	–	–	24.2
... equative	16.5	36.0	29.4	23.2
... difference	9.2	–	–	–
annotated predicates	2778	1327	642	698
... scale arguments	482	130	72	36
... entity E1/E+ arguments	2384	1100	519	661
... entity E2/E- arguments	1775	1070	511	334
... aspect arguments	1851	977	551	505

Table 4.2.: Statistics about our data (IMS), the Jindal and Liu data set (J&L) and the camera (J-C) and car (J-A) parts of the JDPA corpus. The other corpora have been converted to our annotation scheme.

of cameras like the flash or the sensor, or alternatively the pictures produced with the compared cameras.

There are nearly 500 annotations for the argument scale for various predicates. Apart from the expected predicates “*more*”, “*less*”, “*as*”, we have several occurrences of expressions like “*compared*”. There are a few instances where the argument contains a multiword predicate that had to be split up, e.g., in the sentence “*it puts the others to shame*” the annotator wanted to chose “*put to shame*” as the predicate. As we do not allow gaps in predicates and she did not want to include the entity into the predicate, she chose “*put*” as the predicate and annotated “*to shame*” as the modified “adjective”.

Converting other data sets to our annotation scheme. As described in Section 4.2, there are other data sets that have been annotated with comparisons. To compare our data set to them and to be able to use them in our experiments later, we convert these data sets to our annotation scheme.

The J&L data is already split into sentences and tokenized, so we do not need to do further preprocessing. We extract the annotations and map *entityS1* to E1, *entityS2* to E2, and *features* to aspect. Comparison type 1 becomes our type **ranked**, type 2 becomes **equative** and type 3 becomes **superlative**. Non-gradable comparisons do not have predicates or arguments annotated, so they are ignored for our purposes.

For the JDPA data (J-A and J-C), we do sentence segmentation and tokenization with Stanford CoreNLP⁹ (Manning et al., 2014). Annotations are mapped to the extracted tokens, we ignore annotations that do not correspond to complete tokens. We map the preferred entity *More* to E+, the non-preferred entity *Less* to E-, and *Dimension* to aspect. We assign the type **ranked** to all comparisons where *Same* is set to false, otherwise we assign **equative**. For equative comparisons, we assign E+/E- to the entities according to their sentence order. We ignore cars batch 009 where no arguments of comparative predicates are annotated. An annotated argument may be outside the current sentence, but coreference information for entities has also been annotated. We follow the coreference chain to find a coreferent annotation in the same sentence. If this is not successful, the argument is ignored.

To have a uniform and comparable representation of multiword predicates, we do some heuristic splitting of the annotations found in the other data sets into predicate and scale. In both data sets the function word is annotated as the comparative predicate and the content word as the aspect. For every annotated predicate that matches our function word keywords, we check if the token directly following the predicate is annotated as

⁹<http://nlp.stanford.edu/software/corenlp.shtml>

the aspect. If the predicate is “*as*”, we change the annotation from aspect to scale. For the other function words we change the annotation only if the word in question is an adjective (as determined by the Stanford POS Tagger). This serves to distinguish “*less sturdy*” which is a comparative form of “*sturdy*” and thus a multiword predicate, from “*less noise*” where the noun “*noise*” should be the aspect, not part of the predicate. In the J&L data the complete phrase “*as X as*” is annotated as the predicate. We check if the first and last word of a predicate is “*as*”, and take the words in between as scale. The resulting annotations for JDPA and J&L are a bit noisy, but manual inspection shows that nearly all of the scale annotations are correct. We miss some occurrences of multiword predicates in cases where some other aspect is present and has been annotated instead of the content word.

Comparison to other data sets. Table 4.2 contains statistics about the other data sets in the corresponding columns. In the IMS data the percentage of sentences that contain an annotated comparison is higher than in the other data sets: 13% of sentences are annotated as containing at least one comparison, compared to between 8% and 10% of sentences in the other data sets. In all data sets, most sentences contain only one comparison (up to 93% in the J&L data, about 80% in the other data sets). On average, a sentence contains between 1.07 (J&L) and 1.27 (IMS, J-C) comparisons. In all data sets, the majority (50–70%) of comparisons is ranked. In general, annotations for an aspect and an entity are present. Entity 2 is dropped rather frequently in the J&L data.

In total, our corpus contains more than twice as many sentences as the J-A data and roughly three times as many as the J&L and J-C data. This makes it the largest resource dedicated to comparisons in user generated content currently available.

4.5.3. Discussion

During the annotation of our data and while analyzing the other corpora, we have encountered some more general issues and open questions, which we shortly highlight in the following. We focus on the following issues for this discussion: The definition of what is a comparison and what is not (in three different flavors), aspects versus entities, and discontinuous arguments.

Definition of a comparison. Looking at our experiments on Amazon Mechanical Turk and our agreement study, we can confirm what previous work has already discussed: that the decision of what is a comparison and what is not is difficult even for humans.

There are of course many sentences that are very obviously comparisons (e.g., those from the introductory examples) or non-comparisons (e.g., “*I bought A last week*”, “*B is fantastic*”). Some other categories of sentences that look like comparisons but are not (e.g., idioms, correlatives, see also Sentence 2.19 in Section 2.5.1) may confuse annotators at first, but can be excluded with clear annotation guidelines. But even after this, there are still some questionable items:

- (4.12)
- a. “*this is my first digital camera.*”
 - b. “*The 2008 Subaru Impreza WRX STI is based on the Impreza WRX hatchback . . .*”
 - c. “*The images were great both indoors and out.*”
 - d. “*. . . although I would like to see it a little faster.*”
 - e. “*It mirror flip doesn’t sound like a mechanical camera . . .*”
 - f. “*I had to compare this camera with the Nikon D80.*”

Sentence 4.12a is from the J&L data and many similar sentences with the predicate “*first*” have been annotated in this data set. Sentence 4.12b from batch 5 of the JDPA cars data set has the word “*based*” annotated as the comparative predicate. In each case, the other data set does not annotate similar sentences, nor do we in our data. Sentences 4.12c to 4.12f are from the set we used to calculate annotator agreement, which we did after training our annotators and after they had already gained some annotation experience. Sentence 4.12c is a comparison between different usages of the same product, Sentence 4.12d is a wish, Sentence 4.12e is a description of the sound the camera makes. Sentence 4.12f states that there is going to be a comparison, but this is not (yet) it. None of these sentences compare two entities and none should be annotated as a comparison. We have updated the annotation guidelines to explicitly give examples of these categories of sentences to be excluded, but there are probably more such cases that need to be specified.

Limitations of the annotation scheme. Now let’s turn to the other side of the coin, statements that are comparisons, but cannot be annotated in the current scheme. Our working definition of a comparison has been “any statement about the similarity or difference of two entities”, which would include the following examples from our data:

- (4.13)
- a. “*. . . it simulates iso 100 whereas my D70s only did iso 200.*”
 - b. “*Although it yields nearly the same image quality as the D200, it’s unfortunately twice as expensive, lacks environmental seals, . . .*”
 - c. “*It looks closer to film than any digital camera I’ve owned.*”
 - d. “*Pricewise, the H2 slots in between Canon S3 IS and .*”
 - e. “*The 300D and XT are still cheaply made, whatever you say.*”

As two examples that do not fit the current model, Jindal and Liu (2006b) have already discussed juxtapositions (Sentence 4.13a) and non-gradable comparisons of an aspect (here: “*environmental seals*”) that one entity has and the other does not (Sentence 4.13b). In order to annotate these examples, we would need two different aspect slots. Entities can be problematic as well. Sentence 4.13c does compare two entities (“*it*”, “*any digital camera*”), but there is third entity (“*film*”) involved as well. Sentence 4.13d even more clearly describes a relation between three entities (“*H2*”, “*Canon S3 IS*”, [missing token]). This type of relation cannot be put into our current two-entity model and we would need additional comparison types to describe the relations as well. Finally, sometimes the problem is the choice of predicate. Sentence 4.13e expresses an equative comparison between the two cameras in the aspect “*cheaply made*” and would be easy to annotate if it contained the word “*both*”. As it is, there is no good choice for a predicate. It is unclear which of the above examples should be included and in what way. For now, we can only note that the current annotation scheme is not able to capture all possible statements of similarity or difference of two entities.

Implied comparisons and sentiments. It has been noted that every value statement about an entity contains an implicit comparison to some sort of internal standard (Huddleston, 2002). People only note that something is “*good*” or “*bad*” because they compare it against some sort of general standard or expectation about how things should be. This implicit comparison is more overtly present when an adjective is used, e.g., in constructions like the relative “*X is very good*”, the excessive “*X is too good*”, or the assertive “*X is good enough*”. While we certainly would not want to treat every sentiment expression as an implicit comparison, in some cases it may be warranted:

(4.14) “[*D70*]_{E1} *beats* [*EOS 300D*]_{E2} in almost [*every category*]_A, EXCEPT ONE.”

While the comparison at the predicate “*beats*” is annotated, one might argue that there is another comparison implied in the second part of the sentence, namely that the “*EOS 300D*” is at least as good or maybe even better than the “*D70*” in one category (which presumably is further elaborated on in the following sentences of the review). While this is certainly an interesting point for future work, it seems hard to formulate clear guidelines of when such implicit comparisons should be considered and how exactly they should be annotated.

Aspect versus entity. Conceptually, a comparison contains two entities that are compared either in their totality or in some shared aspect. In practice, the distinction between entity and aspect is sometimes hard to make, because aspects form a hierarchy

and aspects of aspects are compared in arbitrarily deep nestings. Consider the following example sentences (annotations from the JDPA camera data):

- (4.15) a. “[*This camera*]_E ... *its* [*screen*]_A *is much bigger than the* [*400D*]_E.”
 b. “... *its* [*screen*]_E *is smaller* [*than the* [*ones*]_E *on some competing models* ...”

The two sentences are very similar, but the annotators took different stands on whether “*screen*” is an entity or an aspect. The annotation of “*this camera*” as an entity allows us to connect it back to some product and list the comparison under its aspect “*screen*”. The annotation of “*screen*” as an entity on the other hand defines what is syntactically compared (the camera’s screen, not only the camera) and allows us to annotate comparisons of aspects of aspects in a parallel way:

- (4.16) “... *its* [*screen*]_E *has smaller* [*resolution*]_A *than the* [*ones*]_E *on* ...”

In our data, we instructed annotators to annotate syntactic entities, i.e., “*its screen*” for the example sentence, and additionally annotate the type of entity as “aspect”. This allows us to treat arbitrarily deep aspect hierarchies and still retain the information that what is syntactically the compared entity in this sentence, is conceptually an aspect of the product we are interested in. Still, even with these guidelines the decision is sometimes hard to make in practice, especially if no other aspect is present in the sentence, and we have discussed many of these cases during annotation. Additionally, the annotation scheme does not capture information about the internal structure of “*its screen*”, where “*its*” refers to the product and “*screen*” to the aspect of the product.

While for training a system and detecting the components, ultimately the exact annotations may not matter so much, the question is very relevant when the detected comparisons are used for aspect-based summaries that group sentiment analysis results for every entity by the recognized aspects. Linking back every annotated entity to some predefined hierarchy of products and their aspects could solve the problem of unambiguously identifying what aspect is talked about, but introduces other problems, namely that we need to create such a resource and that we limit the annotators to the aspects that are contained in this resource. Alternatively, post-processing is necessary in order to be able to generate usable aspect-based summaries from the annotations we have.

Discontinuous arguments. We have discussed multiword predicates and introduced the argument type “scale” for the purpose of modeling them, but arguments can also consist of multiple words that do not always have to be adjacent. Consider the following example from our data:

- (4.17) “*Realistically* [*Micro-cam batteries*]_{E1} (*since they must be very small*) *can’t* [*store*]_{A?} *as* [*much*]_S [*power*]_{A?} *as* [*larger batteries*]_{E2}.”

In the example, “*power*” has been annotated as the aspect, but in reality the discussed aspect is “*store power*”, as in contrast to “*needs power*”, “*provides power*” or any other possible aspect of “*power*”. In our annotation scheme it would have been possible for the annotators to annotate two aspects for one comparisons, but they correctly did not do that as conceptually the annotation of two aspects is intended to denote that the entities are compared in two different, independent aspects, such as in this example:

(4.18) “*However [it]_{E1} is **better** in so many ways such as [image quality]_A and [handling]_A . . .*”

It remains an open question how frequent such discontinuous aspects are and whether it is worth the effort of defining more complex and more flexible representations of comparisons to deal with them.

4.6. Summary

In this chapter, we presented a dedicated gold standard corpus of comparison sentences from English camera reviews. For our purposes we define a comparison as any statement about the similarity or difference of two entities which covers a wide variety of expressions. For each sentence we have annotated detailed information about the comparisons it contains: The *comparative predicate* as the anchor that introduces the comparison, the *type* of the comparison, the two *entities* that are being compared with their *entity type*, and the *aspect* they are compared in. We have described our annotation process and given an overview of our annotation guidelines. The results of our agreement study showed that the decision whether a sentence contains a comparison is difficult to make even for trained human annotators. Once that decision is made, we can achieve consistent results for detailed annotations. In total, we have annotated 2700 comparisons in nearly 2200 sentences from camera reviews which makes our data the largest resource of comparisons in English reviews currently available. The annotations and guidelines are publicly available on our website¹⁰.

¹⁰<http://hdl.handle.net/11022/1007-0000-0000-8E72-0>

5. Detecting comparisons in product reviews

The general approach presented in this chapter has been published in (Kessler and Kuhn, 2013), the sentence and task context experiments in (Kessler, 2014) and the design context experiments in (Kessler and Kuhn, 2014b).

5.1. Introduction

Now that we have a corpus of annotated product comparisons at our disposal, as described in the previous chapter (Chapter 4), this chapter addresses the automatic detection of comparisons and their components with machine learning methods. We need methods that differ from standard aspect-oriented sentiment analysis methods, because most standard approaches in sentiment analysis address sentiment expressions assign one polarity to one target entity. This is insufficient for comparisons which involve more than one target entity and may involve the assignment of more than one polarity. It is thus necessary to analyze comparisons separately, but the analysis of such comparisons has not received a lot of attention in the sentiment analysis community (see Section 2.5 for an overview).

Comparisons are relatively frequent in product reviews (5%-10% of sentences in our data). Some comparisons are among competing products as a whole, most compare a certain aspect of the two products. Comparisons are of interest for companies that do not only want to know what aspects of their product users like or dislike, but also where they stand in relation to their competitors. Also, comparisons are presumably the most useful kind of expression when it comes to supporting a process of choice, as they provide explicit comparative judgments which may enable the user to make a decision between several products. Comparisons are also less influenced by confounding factors like cultural and personal differences in expressing sentiment than other sentiment expressions. In some cultures “*not bad*” may be the highest praise, and in some other

culture the expression “*awesome*” may indicate rather average performance. But when two entities are compared, an explicit ranking is given that cannot be misinterpreted.

A comparison contains several components that must be identified in order to get meaningful information (cf. Chapter 4). Consider the following example:

(5.1) “[*It*]_{E1} had a **better** [*lens*]_A than [*my old camera*]_{E2}.”

In this example, the anchor for the comparison (the *comparative predicate*) is the word “*better*”. The sentence compares two *entities*, in this case two cameras referenced by “*It*” and “*my old camera*”, and they are compared in the *aspect* “*lens*”. A ranking between the entities is introduced, where the first entity is preferred over the second. The task of *comparison detection* is to identify all these components in order to capture the full relation between the two entities.

Given a sentence that contains a comparison, we train a machine learning system on our data to detect the relevant components. We follow the work of Hou and Li (2008) on Chinese comparisons and use a methodology inspired by semantic role labeling (SRL). Semantic roles describe the relation of participants to an event (e.g., agent, patient, topic), abstracting over the different possible surface realizations of the same semantic structure (Fillmore, 1968). An event is expressed by a predicate and the participants are expressed by arguments of the predicate that fill the different semantic roles that are defined for the given predicate. SRL predicates are often verbs, but the framework is more general and allows predicates of other parts-of-speech as well.

Adapted to the task of comparison detection, the “events” we are interested in are comparative predicates that introduce a comparison, and the “participants” are the two entities and the aspect that is being compared. The following example shows the SRL annotation from PropBank for an example sentence with the predicate “*gave*” (frame 01 of “*give*”) and its three arguments (A0, A1, A2) and contrasts it with the annotation of a comparison sentence with the predicate “*better*” and its three arguments (E1, E2, A):

(5.2) a. “[*He*]_{A0} **gave**_{give.01} [*her*]_{A2} [*the book*]_{A1}.”
 b. “[*It*]_{E1} had a **better** [*lens*]_A than [*my old camera*]_{E2}.”

The annotations for the two tasks are very similar, so it seems reasonable that similar methods can be successful. In putting ourselves inside the framework of SRL, we can take advantage of the existing methods and systems that have been developed and tested for this task since its introduction by Gildea and Jurafsky (2002). The detection of comparisons is in many aspects similar to SRL in the challenges we expect to encounter, e.g., SRL has dealt with multiword predicates, implicit arguments and sparsity of training data for individual predicates. On the other hand, we also expect to find

some differences. For SRL, the possible predicates, the variety of frames they can express and the arguments they can take are standardized in large resources such as PropBank (Palmer et al., 2005) or FrameNet (Baker et al., 1998). Such resources do not (yet) exist for comparisons and due to the diversity of possible ways of expressing comparisons and the large number of possible entities to be compared, we expect the predicates and arguments in our task to be more heterogeneous categories than in standard SRL. As such, lexicalized features that have worked well in SRL may not be as helpful for our task. As an additional, orthogonal challenge, we work on user generated data, while most of the work on SRL has been done on news data. The non-standard grammar and spelling which is often used in user generated content makes tasks such as part of speech tagging or parsing more difficult on this type of data than on news data.

Transferring an SRL-inspired method to another task is not uncommon. SRL-like approaches have been used for event graphs (Glavaš and Šnajder, 2015) where events consist of an predicate anchor (e.g., a verb like “*kill*”) and the participants or circumstances (agent, time, location) are the arguments; or recipes (Malmaud et al., 2014) where events are actions that affect state transformation (e.g., “*stir*”) and arguments are the ingredients and other information (e.g. “*add water*”, “*stir for 5 minutes*”).

SRL systems have also previously been applied to sentiment analysis. Kim and Hovy (2006) use a semantic role labeler that associates opinion words with semantic frames from FrameNet and then use manually defined mappings from the identified frame elements to identify the target and holder of the sentiment expression. Later, Ruppenhofer et al. (2008) discuss semantic role labeling in the context of sentiment analysis. While they present examples for cases that require knowledge beyond semantic role labeling, they argue that SRL can still make an important contribution to the recognition of targets and holders of opinions. In follow-up work, Ruppenhofer and Rehbein (2012) propose SentiFrameNet, constructed on top of FrameNet, which adds opinion frames to sentiment predicates with slots for the opinion holder, target, polarity and intensity of the sentiment expression. Finally, closest to our approach, Hou and Li (2008) have worked on comparison detection for Chinese with an SRL system, but without investigating the role of context in detail.

This chapter presents our system CSRL (Comparison Semantic Role Labeler) for the automatic detection of comparative predicates and their arguments. We build on the SRL system by Björkelund et al. (2009) and re-train it for our task on the comparison data presented in Chapter 4. Like semantic roles, comparisons are realized in a structural way, so we expect methods that look only at isolated words to have a poor performance and anticipate that context is needed to reliably identify predicates and arguments.

Context can be added in many different forms, on different levels and in different ways, out of which we only look at the part that seemed the most promising. Specifically, we look at three types of context: *sentence* context, context from knowledge about the *task* and the domain, and context from the *annotation design* of the data used for training our machine learning system. These are not the only possible forms of context, other possibilities include discourse context or context about the products that are reviewed, which may provide interesting future work.

For the experiments on sentence context, similar to what we did for polarity reversers in Chapter 3, we use dependency tree information for sentence context and compare to a window-based context used in previous work (Jindal and Liu, 2006b). We hypothesize that structural information from a dependency tree is more helpful than window-based context information, even if the parses on our user generated data are noisy. To include context from the task and domain, we use generalization techniques to overcome sparsity issues, add information about possible types of comparisons, and include sentiment polarity information of words in the analysis. Context from the annotation design refers to the decisions taken at annotation time about the linguistic anchoring of comparisons and the possible argument types to chose from, specifically we focus on the annotation of multiword predicates and on the labels of arguments. We systematically change those annotations in our data and analyze the influence of the changes on the classification performance of our system.

The main research question we are going to address in this chapter is thus an adapted version of research question A:

Research Question A2: *How can structural linguistic context information based on a dependency parse tree be used for the reliable detection of comparisons and their components?*

Specifically, this chapter explores the following questions:

- Can comparison detection be successfully performed by our system CSRL, an SRL system re-trained for the new task on our comparison data?
- Are features based on structural context information from a dependency parse more helpful than those from window-based context information?
- Is it beneficial to include context information from the task and domain?
- How do annotation design decisions influence the performance of our system?

5.2. Approach

The task we want to solve for a given comparison sentence is to detect the comparative predicate, the entities that are involved and the aspect that is being compared. We borrow our methodology from semantic role labeling (SRL) and employ a similar pipeline approach. The input to our system which we call CSRL (Comparison Semantic Role Labeler) is a sentence that we assume to contain at least one comparison. The result of our processing are one or more comparative predicates per sentence and their arguments.

In the following we describe our system in more detail and discuss the different forms of context that we investigate.

5.2.1. Using an SRL approach to detect comparisons

To detect comparisons, we use a standard pipeline approach from SRL shown in Figure 5.1. As a first step, the comparative predicate is identified. This is a binary classification decision for each token of whether or not it is a predicate. In our case, the predicates we are looking for are not predicates in the sense of semantic role labeling, but comparative predicates, e.g., “*better*”, “*superior*”. The next step in SRL would be predicate disambiguation to identify the different frames this predicate can express. As we do not have such frame information, no predicate disambiguation performed.

After we have identified the predicates, the next step in the pipeline is to identify for each predicate the arguments. The identification step is again a binary classification of whether a word in the sentence is an argument of the given predicate. Arguments are always related to a specific predicate, so if a sentence contains more than one predicate, the identification of arguments is done for each of the predicates separately. As a final step in the pipeline, the role or type of each found argument is classified. In our case of comparisons, we distinguish the two entities, the aspect and, in case of multiword predicate, the scale (for details see also the description of our data in Chapter 4).

We use an existing SRL system (Björkelund et al., 2009)¹ that is part of the MATE toolkit and re-train it on our data for the new task. For all classification steps, the system uses regularized linear logistic regression from the LIBLINEAR package (Fan et al., 2008). We set the SRL system to train separate classifiers for predicates of different parts of speech (POS), but consider all possible POS. In preliminary experiments, we have found this to perform slightly better than training one classifier for all kinds of predicates, although the difference is not significant. We do not use the reranker. In

¹<http://code.google.com/p/mate-tools/>

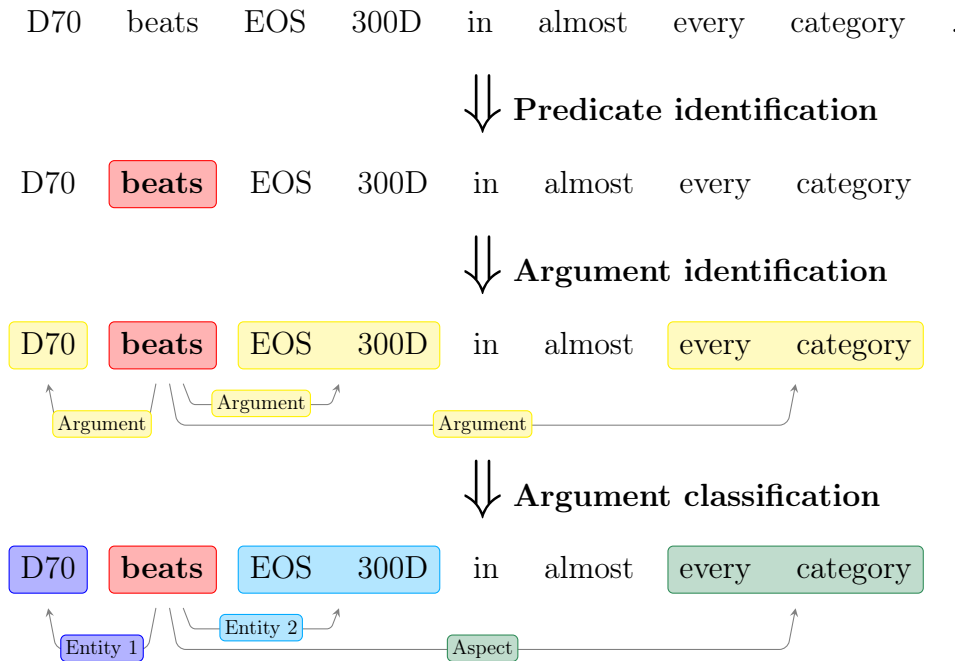


Figure 5.1.: Steps of the comparison detection pipeline for an example sentence.

general, our approach is not tied to this particular SRL system or a particular parser.

The features of the system are based on the output of the MATE dependency parser (Bohnet, 2010). On a basic level, for every token we can extract word features such as the surface form, lemma and POS. As arguments are always extracted for a specific predicate, for argument candidates, the word features can also be extracted for the current predicate. Another basic feature is the relative position of the candidate argument with respect to the predicate (before, after, same). Features that model context will be introduced in the following subsections.

5.2.2. Sentence context

The first type of context we investigate is context from the sentence containing the comparison. As comparisons are a complex verbalization, we think that context is necessary for the reliable identification of predicates and arguments. We compare adding sentence context in two different ways that are commonly used in NLP: with a window-based approach, and structural linguistic context from a dependency parser. Figure 5.2 shows the added context information for an example argument candidate.

The most simple way to add sentence information is to include the context words before and after the candidate predicate or argument. We add word features for each of the context words in a symmetric window to both sides (left and right) of the candidate

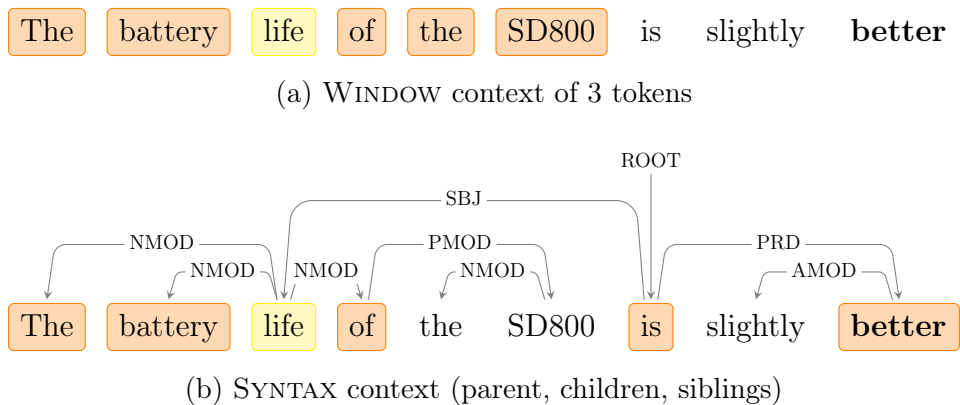


Figure 5.2.: Sentence context around the argument candidate “*life*” for the predicate “*better*”. Word features are added for the words in frames.

word. We call this WINDOW context.

Alternatively, we can parse the sentence and use information from the syntactic structure as features. We use the MATE dependency parser (Bohnet, 2010) to parse a sentence which returns the structure as a set of parent-child relations between the words in the sentence. The parser is trained on news texts, but applied to texts from blogs and reviews. These texts contain errors in spelling, grammar and punctuation, so the parses may not always be correct. Incorrect parses might lead to confusing information for the classifier. It remains to be seen whether these parse features are helpful even in these circumstances. The SYNTAX context adds word features from the parents and children of the predicate candidate. For arguments, in addition to the features from the predicate, children and siblings of the argument and the dependency path from argument to predicate are added to the feature set.

5.2.3. Context from knowledge about the task

There are several other ways of including context in the form of background knowledge about the task and the domain. In a second set of experiments we investigate the effect of different types of context which use this sort of information.

First, we investigate the use of generalization techniques to overcome sparsity issues. Taking advantage of our knowledge about the entities in the domain (cameras), we add a preprocessing step that replaces typical camera model references (number-letter combinations like “*EOS 200D*”) with a placeholder to generalize over the multitude of different camera models. This preprocessing step may also help to improve the parse trees of sentences, as product names are not treated well by the parser. We call these

configuration NER. A more universal generalization feature over word classes uses Brown clusters (Brown et al., 1992) as features to enhance detection of rare predicates or aspects. We call this configuration CLUSTER. We expect both generalization methods to have a positive influence on all steps of the pipeline.

The types of comparisons differ in their typical usage of arguments. For example, superlatives often drop the second entity for pragmatic reasons, but ranked comparisons usually take two explicit entities. We implement a feature TYPE that models the comparison type with four values: ranked, superlative, equative and non-graded difference (cf. Section 4.4). This feature is applicable to argument identification and classification. We use gold information for our initial experiments to see whether the performance gain is worth investing time in the automatic classification of comparison types.

Lastly, we see that the distinction between the preferred and non-preferred entity is dependent on the sentiment expressed by the predicate:

- (5.3) a. “[It]_{E+} is **positive-JJR** than the [XYZ]_{E-}.”
 b. “[It]_{E-} is **negative-JJR** than the [XYZ]_{E+}.”

We add the feature POLARITY to model the prior polarity from a sentiment dictionary (positive, negative or neutral) of the predicate. This feature can be added to all steps of the pipeline, but we expect it to be useful especially in argument classification when a distinction is made between entities according to their ranking.

5.2.4. Context from annotation design

Context from annotation design refers to the annotation design decisions we have made in creating our corpus as described in Chapter 4. We focus on three specific questions, the first two concern the annotation of multiword predicates like “*less good*” or “*as good as*”, the third concerns the annotation of entities. In our data (and to a certain degree also for the other existing resources) we can systematically change the annotations to reflect a different outcome for each decision and investigate the effect these changes have on the classification performance of our system.

Anchoring of multiword predicates. Most comparative predicates are single words like “*better*” or “*best*”, but multiword predicates account for about 10-20% of comparative predicates in our data. Some are expressions like “*X has the edge over Y*” or “*X is on par with Y*” which we will not discuss in this work. But the majority of multiword predicates in our data are systematically introduced by English grammar rules. Consider the following variations of a sentence:

- (5.4) a. “[It]_{E1} had a **sturdier** [feel]_A than [X]_{E2}.”
 b. “[It]_{E1} had a **less sturdy** [feel]_A than [X]_{E2}.”

Sentence 5.4a compares the aspect “*feel*” of a camera to some other camera with the comparative predicate “*sturdier*”. If we change the direction of the comparison, we get a multiword predicate with the modifier “*less*” added to the adverb (Sentence 5.4b). In the following we refer to such a modifier as *function word* and to the modified adjective or adverb as *content word*. Besides the modifiers “*less*” and “*more*” for comparative forms, and “*most*” and “*least*” for the superlative, the list of function words includes “*as*” which is used to introduce an equative comparison like “*X is as good as Y*”.²

Our system, like the majority of systems to date, uses a single-token-based approach for the automatic detection of comparative predicates which raises the question of which word to select as an anchor for multiword predicates. A strong argument can be made to select the function word as the token anchor for the comparative predicate. There will be more training instances to use in machine learning for a given function word than for the individual content words, so sparsity is reduced. On the other hand, choosing the content word may be more informative for end users.

The first question we want to investigate in this study is whether the different annotation decisions translate into a difference in classification performance. In our first experiment we identify all occurrences of multiword predicates. In one setting (FUNCTION predicates), we annotate the modifying function word as the comparative predicate. In the second setting (CONTENT predicates), we annotate the modified content word. The following illustrates the different annotations for an example sentence:

- (5.5) a. “... had a **less** [sturdy]_A [feel]_A ...” (FUNCTION predicates)
 b. “... had a **less** [sturdy]_A [feel]_A ...” (CONTENT predicates)

In both cases we have the same number of comparative predicates, only the annotations differ. Argument annotations are identical, even if it causes one token to have a predicate and an argument annotation at the same time.

Annotation of aspects and scale. The second question deals with the annotation of the content word when we use function predicates for multiword predicates. Most existing corpora annotate the content word as an aspect. We will illustrate some problems with this approach with the following examples:

- (5.6) a. “... a **sturdier** [feel]_A ...”

²Note that not all occurrences of the keywords indicate multiword predicates, e.g., in “*X has less noise*” the word “*noise*” is not part of the predicate but the compared aspect.

- b. "... a *less* [*sturdy*]_A [*feel*]_A ..."
- c. "... a *less* [*sturdy*]_A *feel* ..."
- d. "... a *less sturdy* [*feel*]_A ..."
- e. "... a *less flimsy* [*feel*]_A ..."

If we compare the annotations of Sentence 5.6a and Sentence 5.6b we see that changing the direction of the comparison introduces a new aspect. This is counter-intuitive because what is compared (i.e., the aspect) should not depend on the introduced ranking. Additionally, if there is only one slot for the aspect, as is the case in one of the corpora we use, annotators will need to decide between annotations 5.6c and 5.6d. Annotation 5.6c is inconsistent when compared to annotation 5.6a as only the direction of the comparison has changed, but as a result we have different annotations for the aspect. With annotation 5.6d we lose information about the actual sentiment polarity that is expressed, as we are not able to distinguish it from the annotation in Sentence 5.6e.

To solve these issues, we have proposed to introduce a separate argument called scale with the sole purpose of modeling the content word in a multiword predicate (cf Section 4.4). In our second design context experiment, we use function words as predicates and change the label of the content word from aspect (ASPECT) to scale (SCALE). This results in the following annotations being compared:

- (5.7) a. "... had a *less* [*sturdy*]_A [*feel*]_A ..." (function predicates with ASPECT argument)
 b. "... had a *less* [*sturdy*]_S [*feel*]_A ..." (function predicates with SCALE argument)

The tasks of predicate and argument identification are independent of argument labels, so the only change is in argument classification. We expect a drop in classification performance due to the increased number of classes. We hope that the drop is not significant, as the new argument class is well-defined and should be relatively easy to distinguish from real aspects.

Annotation of entities. Our third question deals with the annotation of entities. Usually, two entities participate in a comparison. It is conceivable to treat both entities as one type of argument and to not differentiate between them (UNIFIED entities). When two different types of entities are distinguished, there are two possibilities. Entities can be annotated according to surface position as entity 1 and entity 2 (SURFACE entities), or by preference as a preferred and a non-preferred entity (PREFERENCE entities). Thus, the following setups are compared:

- (5.8) a. "[*It*]_E had a *less* [*sturdy*]_A [*feel*]_A than [*the other one*]_E." (UNIFIED entities)
 b. "[*It*]_{E1} had a *less* [*sturdy*]_A [*feel*]_A than [*the other one*]_{E2}." (SURFACE entities)

c. “[It]_{E-} had a *less* [*sturdy*]_A [*feel*]_A than [*the other one*]_{E+}.” (PREFERENCE entities)

For entities where there is no ranking introduced (equative comparisons), we use surface order also in case of PREFERENCE entities. Like for the previous question, the different annotations only affect the argument classification step.

5.3. Experiments

5.3.1. Data and experimental setup

Data. The main data for our experiments is the set of camera reviews described in Chapter 4, referenced with IMS. We also compare to the two other published resources described in more detail in Section 4.2: the camera (J-C) and car (J-A) parts of the JDPa corpus (J. Kessler et al., 2010) and the J&L data (Jindal and Liu, 2006b)³. The data is converted to our annotation scheme as described in section Section 4.5.2. For the experiments that do not involve annotation design decisions, we map the annotations of all entities to one argument type *entity* (i.e., UNIFIED entities) and we map the scale argument that exists only in our data to *aspect* (i.e., ASPECT configuration).

Dependency parses are produced by the MATE dependency parser (Bohnet, 2010)⁴. Prior polarity for the POLARITY feature is taken from the MPQA list of subjectivity clues⁵ (Wilson et al., 2005). CLUSTER features are based on 500 clusters created on news data over all words that occur at least 25 times.

Evaluation. We evaluate on each data set separately using 10-fold cross-validation. Folds are created through rotation by putting sentence i in the i th fold. We report precision (P), recall (R) and F_1 score (F). For argument classification, scores are micro-averaged over all argument types (P_μ , R_μ , F_μ). For argument identification and classification, we report results on gold predicates, to exclude the propagation of errors in predicate identification.

Statistical significance is measured with the approximate randomization test (Noreen, 1989). We use 10000 iterations and consider a difference in F_1 score statistically significant at $p < .05$. In cases where we cannot calculate significance because annotations change between experiments, we report the absolute differences in F_1 score (ΔF).

³In their publication, Jindal and Liu (2006b) work on some unknown subset of the J&L data, so our results are not directly comparable to the results reported there

⁴<http://code.google.com/p/mate-tools/>

⁵http://www.cs.pitt.edu/mpqa/subj_lexicon.html

Baselines. We have implemented two baselines for predicate identification based on previous work. The simplest baseline, POS, classifies all tokens with a comparative part of speech as predicates (comparative and superlative adjectives and adverbs, i.e., the POS tags 'JJR', 'JJS', 'RBR', 'RBS'). A more sophisticated keywords BASELINE, uses a list of about 80 manually compiled comparative keywords and keyphrases from (Jindal and Liu, 2006a) in addition to the words identified by POS.

No previous work has presented a baseline for arguments. Our BASELINE for argument identification and classification, uses some heuristics based on the characteristics of our data. Most entities are nouns or pronouns, so we mark the first noun or pronoun before and after the predicate as entities. If different types of entities are distinguished, we mark the entity before the predicate as entity 1 or E+, the entity after the predicate entity 2 or E-, as most predicates have positive sentiment. If the predicate is a comparative adjective, we classify the predicate itself as aspect, because this type of annotation is very frequent in the JDPA data. For other predicates except nouns and verbs, we annotate the direct head of the predicate in the dependency tree as aspect.

SRL system. We use the MATE Semantic Role Labeling system (Björkelund et al., 2009)⁶ with default settings and without the reranker. We re-train the system on our data sets to identify comparative predicates and arguments. We perform three classification steps: predicate identification, argument identification and argument classification. We vary feature sets for the experiments, but always add the same type of feature to all steps and use the same feature sets for both steps involving arguments.

As a starting point for our context experiments, we use a MINIMAL feature set that includes only word features of the predicate candidate and argument candidates. For the tasks of argument identification and classification, it additionally contains the relative position of the candidate with respect to the predicate (before, after, same).

5.3.2. Results

To remove the influence of errors introduced by the relatively low performance of all systems on predicate identification, we use annotated predicates (gold predicates) for the argument experiments. All results drop about 10% when system predicates are used, but the differences between the systems stay the same. In the following discussion we mainly use the numbers for our own IMS data, but the results for all data sets are shown in the figures. More detailed results can be found in Appendix B.

⁶<http://code.google.com/p/mate-tools/>

Results for the baseline and minimal system

First, we compare CSRL with MINIMAL features to the baselines POS and BASELINE. The first three bars of each plot group in Figure 5.3 show the F_1 scores of these systems for the different data sets and tasks (detailed results can be found in Table B.1).

The keyword BASELINE for predicates significantly outperforms POS with the huge margin of 68.2 vs. 60.7 on the IMS data. This shows that an important percentage of our predicates are not comparative words in the linguistic sense. For predicate identification, the MINIMAL system already gives a significant improvement over both baselines, achieving an F_1 score of 79.6 on the IMS data. The performance increase is the result of words that are missing in the list that are learned to be predicates (e.g., “*improvement*”, “*compare*”), as well as cases where the POS tag is enough to distinguish between the use of a word as predicate (“*like*” as an adverb) and not a predicate (“*like*” as a verb). Here are some example sentences with recognized predicates marked in bold and the signs ✓/ ✗ indicating whether this decision was correct or wrong:

- (5.9) a. “*It feels just like ✗ the Rebel.*” (POS)
 b. “*It feels just **like** ✓ the Rebel.*” (BASELINE, MINIMAL)
 c. “*Image quality is an **improvement** ✗ over the D70.*” (POS, BASELINE)
 d. “*Image quality is an **improvement** ✓ over the D70.*” (MINIMAL)
 e. “*Overall I **like** ✗ the SD600 camera.*” (POS, BASELINE)
 f. “*Overall I **like** ✓ the SD600 camera.*” (MINIMAL)

For argument identification, the MINIMAL system gives results clearly below the baseline (F_1 score of 33.6 compared to 45.3). If we look at precision and recall, we see that MINIMAL suffers mainly in recall due to a very heterogeneous set of different arguments. Numbers in general are not very high and both systems miss many arguments. Consider the following examples where both systems get the basic sentence right, but fail on slight changes of that same sentence:

- (5.10) a. “*The [D200]_E ✓ is still **faster** than this [camera]_E ✓.*” (BASELINE, MINIMAL)
 b. “*... than [my]_E ✗ camera ✗.*” (BASELINE)
 c. “*... than my ✓ [camera]_E ✓.*” (MINIMAL)
 d. “*... than the D70 ✗.*” (BASELINE)
 e. “*... than the D70 ✗.*” (MINIMAL)

While for argument classification the BASELINE still outperforms the MINIMAL system, the performance differences get smaller (32.2 vs. 34.5, only a difference of 2 points compared to a difference of 12 points in argument identification). The low recall of argument identification is partly responsible for the generally low numbers in argument

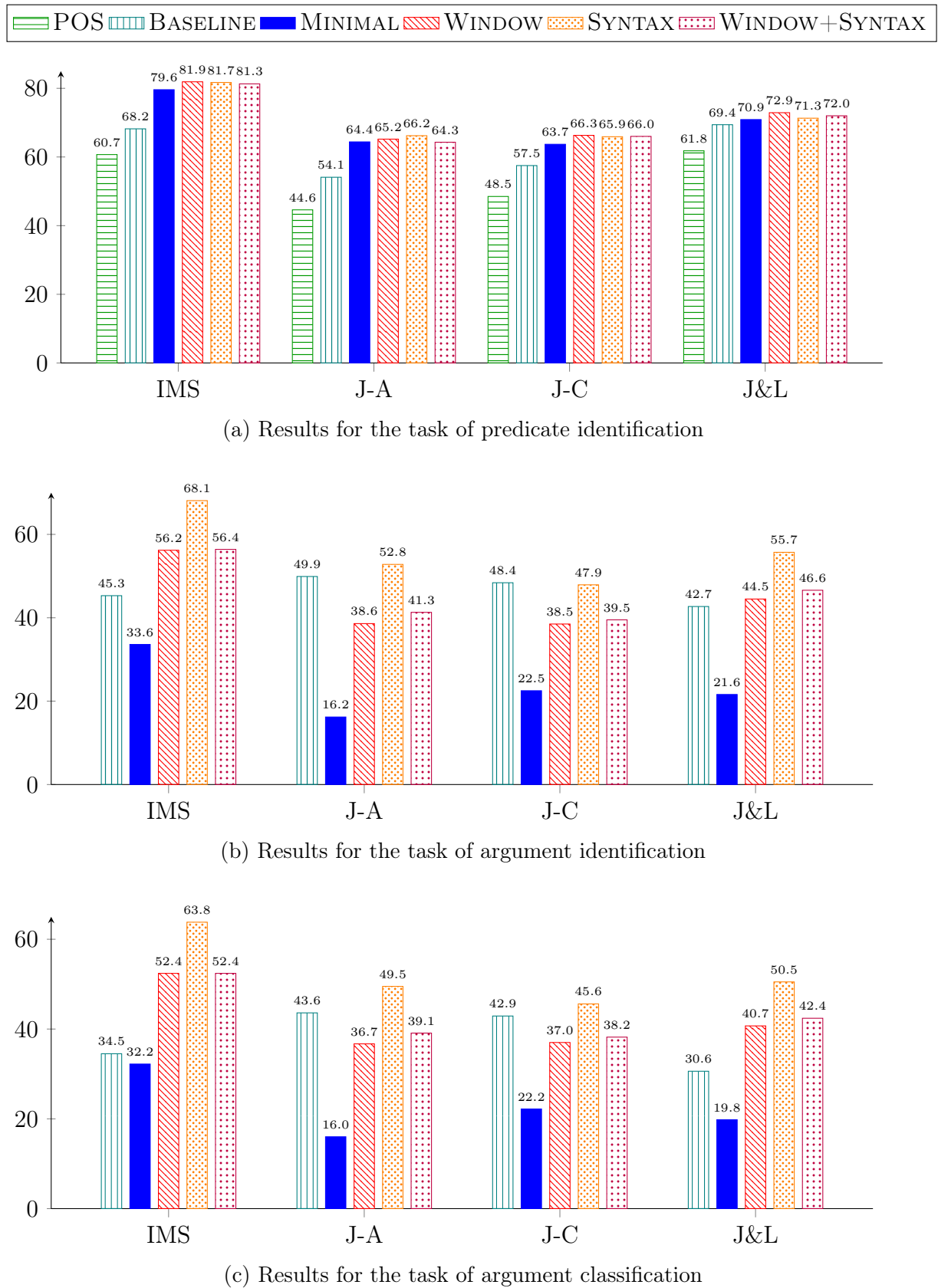


Figure 5.3.: F_1 score results of the experiments on sentence context for the three tasks and the four data sets (IMS, J-A, J-C, J&L).

classification. The following is an example where the system has learned that “*image quality*” is a typical aspect rather than an entity:

- (5.11) a. “[Image quality]_E ✗ is an *improvement* over the D70.” (BASELINE)
 b. “[Image quality]_A ✓ is an *improvement* over the D70.” (MINIMAL)

Results for sentence context

The other bars in Figure 5.3 show some results of adding sentence context to our system (detailed results can be found in Table B.1). WINDOW is the best window-based approach which uses a symmetric window of 2 tokens for predicates and a symmetric window of 3 tokens for arguments. SYNTAX uses all available syntactic information, which we found in our experiments to perform better or comparable to using any subset.

Adding any type of sentence context improves the recognition of predicates (F_1 score of 81.9 for WINDOW and 81.7 for SYNTAX compared to 79.6 for the MINIMAL system). As an example for improved recognition consider these sentences:

- (5.12) a. “The auto white balance is not *as* ✓ good *as* ✗ my D80 was.” (BASELINE)
 b. “The auto white balance is not *as* ✗ good *as* ✓ my D80 was.” (MINIMAL)
 c. “The auto white balance is not *as* ✓ good *as* ✓ my D80 was.” (WINDOW, SYNTAX)

For each comparison there are two instances of “*as*”, the first one a predicate, the second not. Without context, they are either both marked as predicates (BASELINE, the word is in the list of keywords) or neither of them (MINIMAL, the evidence that this is a predicate is at most 50%). Only with context can the system learn to distinguish between the two. There are a few long dependencies where adding SYNTAX is better than WINDOW, e.g., “*as much power as*”, but not enough to impact overall performance. In most data sets the window-based approach is slightly better than SYNTAX, but the differences are not significant.

For argument identification, WINDOW (F_1 score of 56.2) outperforms the baseline (F_1 score of 45.3) significantly by a considerable margin and is in turn significantly outperformed by SYNTAX (F_1 score of 68.1) on the IMS data. Argument classification results show the same order of systems (F_1 scores of 34.5 for the BASELINE vs. 52.4 for WINDOW vs. 63.8 for SYNTAX). Consider the following sentences as illustration:

- (5.13) a. “The [G7]_E ✓ is much *easier* to hold ✗ .” (MINIMAL)
 b. “The [G7]_E ✓ is much *easier* to [hold]_A ✓ .” (WINDOW, SYNTAX)
 c. “The battery life ✗ of the [SD800]_E is slightly *better* .” (WINDOW)
 d. “The [battery life]_A ✓ of the [SD800]_E is slightly *better* .” (SYNTAX)

In the first sentence, “*hold*” is a rather unusual aspect, so only with the context information that “*JJR to*” usually introduces an aspect, the system is able to decide that the word should be an aspect. Similar patterns are found for unusual product names that occur in a clear position for entities. In Sentence 5.13c the distance between the argument and the predicate is very large, so the predicate falls outside of the considered context words for WINDOW. Only with syntax information we can detect that “*battery life*”, a sibling of the predicate, needs to be an argument. In the JDPa data, the result with respect to the baseline for argument identification and classification is different. WINDOW does not manage to outperform the baseline and SYNTAX manages only by a small (but still significant) margin. Part of the reason may be, that we developed our heuristic baseline using the statistics of the JDPa camera data. Still, differences between SYNTAX and WINDOW are clear.

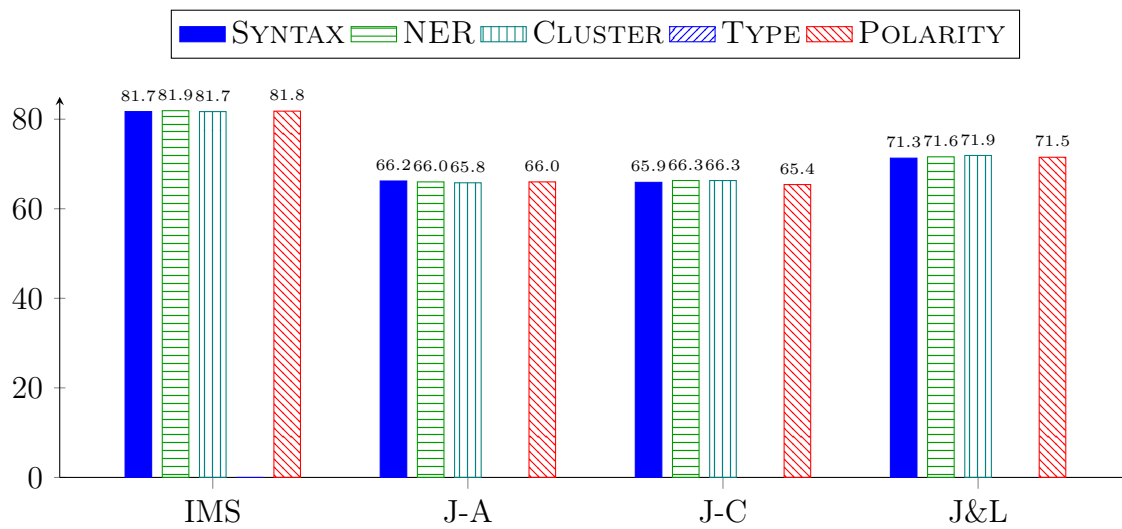
Using both types of context together (WINDOW+SYNTAX) does not lead to a better performance on any task, which indicates that the system learns the same information with both types of context. It is an encouraging result that even in our noisy data structural context from syntax outperforms a window-based approach for arguments.

Results for context from knowledge about the task

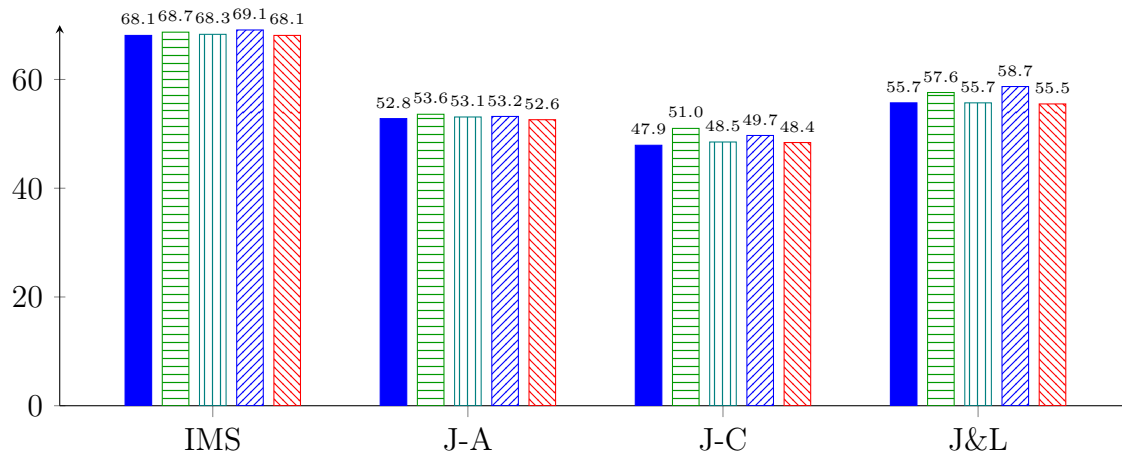
Figure 5.4 shows results for adding different types of task and domain context to the system (detailed results can be found in Table B.4). We add the features individually to SYNTAX, best sentence-context system, to judge their impact.

Generalization with CLUSTER and NER gives mixed results. For predicate identification, we would have expected Brown clusters to be helpful to generalize over different expressions of the same semantic content, but the effect is slight if at all present. Argument identification and classification benefit slightly. A possible reason for the small impact of the cluster features may be our use of general, non-domain specific clusters. Counter-intuitively, the setting NER, designed to help with argument identification, slightly improves predicate identification as well in the IMS data. This is mainly an effect of the better parse trees produced by the parser when we replace long product names by a placeholder before parsing, as the parser is often confused by such product names, e.g., “*Canon EOS 5D Digital SLR*”. Argument identification and classification benefit from NER. Still, overall performance differences are small, so that we did not pursue the issue further.

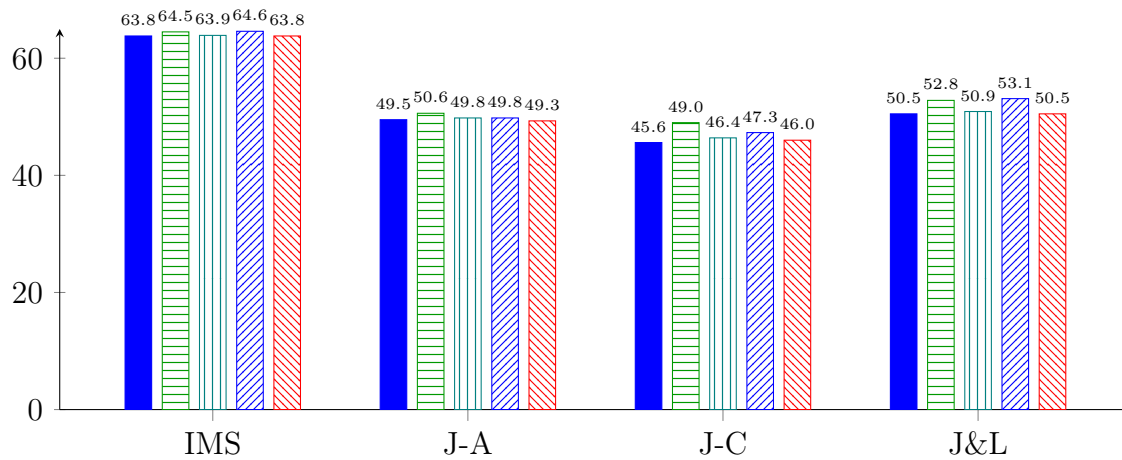
Adding information about the type of the comparison (TYPE) can be done only after the identification of the predicate. For our experiments, we use gold predicate type



(a) Results for the task of predicate identification



(b) Results for the task of argument identification



(c) Results for the task of argument classification

Figure 5.4.: F_1 score results of the experiments on context from task and domain for the three tasks and the four data set (IMS, J-A, J-C, J&L).

	J&L	J-C	J-A	IMS
all predicates	668	642	1327	2778
multiword predicates	36	71	127	338
– <i>more</i>	13	26	68	163
– <i>less</i>	4	6	12	31
– <i>most</i>	2	1	4	14
– <i>least</i>	0	0	1	2
– <i>as</i>	17	38	42	128

Table 5.1.: Statistics about the multiword predicates in the data.

information, but learning to distinguish different comparison types automatically would be feasible. For both argument tasks, adding type information gives a small improvement due to increased recall. The small boost to know that some comparison types usually take an argument is enough to detect more infrequent arguments. The performance differences are significant for both tasks (except argument classification on J-C data), so the automatic detection of comparison types may be a worthwhile task for future work.

Adding sentiment polarity information (POLARITY) is designed to help with argument classification. It can be added to every step of the pipeline, but as expected there are no substantial improvements on predicate and argument identification. The changes in argument classification are also very slight, because the feature is designed to distinguish the preferred from the non-preferred entity and for this set of experiments we are mapping both entities to the same label. We have used this feature on the PREFERENCE entities setting (not shown), but even there POLARITY gives only slight improvements. There are two possible reasons that come to mind. First, the limited coverage of the sentiment dictionary, and second the fact that our feature does not take into account the contextual polarity of the predicate, e.g., modification by negation. The connection of comparisons and sentiment polarity is an interesting question for possible future work.

Results for annotation design context of multiword predicates

We now turn to the experiments involving annotation design decision. The first experiment concern the annotations of multiword predicates. Table 5.1 give some statistics about the number of multiword predicates in our data. We always use the best system for the classification, i.e., SYNTAX. As annotations change between settings, we cannot calculate statistical significance, instead we give the absolute differences in F_1 score (ΔF). For the J&L and the JDPA data, the splitting of multiword predicates has been done automatically, so results are a bit noisy.

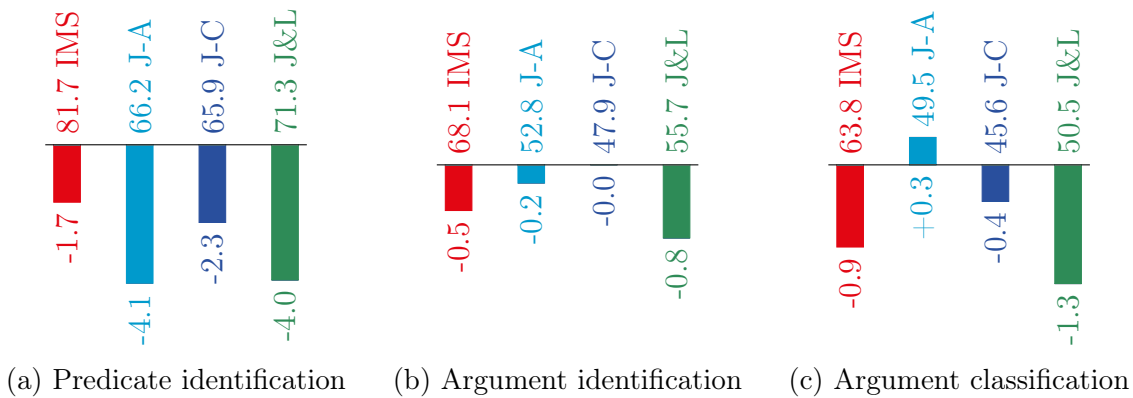


Figure 5.5.: Results of the experiments of changing annotation design decisions for multiword predicate annotations. The number above the bar shows the F_1 score for the FUNCTION predicates setting in the four data sets (IMS, J-A, J-C, J&L). The bars show the difference ΔF to the CONTENT predicates setting.

Figure 5.5 shows some results of the experiments involving annotation design context (detailed results can be found in Table B.6). When we compare the annotation of FUNCTION predicates with CONTENT predicates, we see that annotating the content word decreases performance on predicate identification. This fits our expectation as lexical features have large weight in the learned model and by choosing many different adjectives over few function words we make the data more sparse. The decrease is quite large compared to the relatively small number of changes we are making.

Performance for argument identification and classification also suffers on the IMS data and the J&L data. But the JDPA data sets are not as much affected or even gain in performance (J-A gains 0.3 points on argument classification). Part of this is due to the fact that the content predicate setting over-generates aspects that are the same token as the predicate even for single word predicates like “*faster*”. Such annotations are common in the JDPA data sets, but never occur in the other data sets. The increased recall for aspects then balances the loss on the other arguments.

Results for annotation design context of argument labels

The next experiments concerns only the task of argument classification, so Figure 5.6 shows only the results for this task. For the first setting, SCALE, we use function predicates and change the annotation of the content word from aspect to scale. Entities are unchanged. If we train a model to detect three types of arguments instead of two, we expect a drop in performance simply due to the increased number of classes. But as the new argument class is well-defined and should be relatively easy to distinguish from

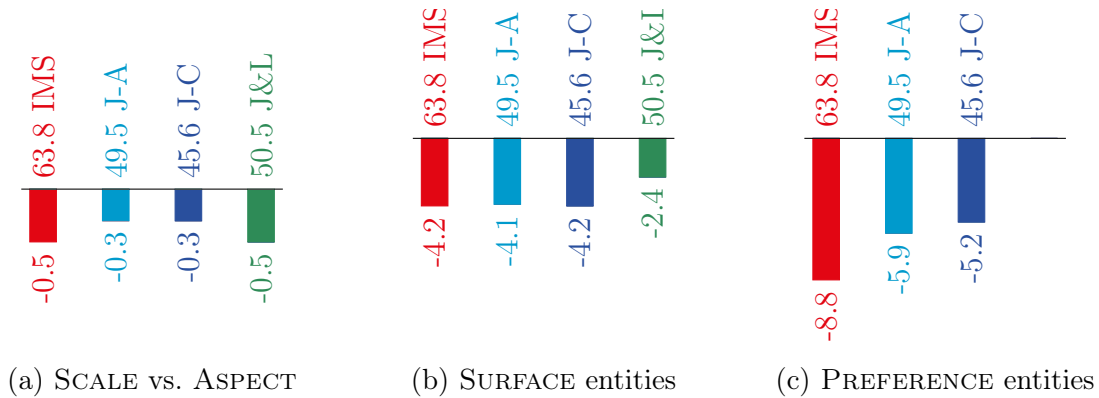


Figure 5.6.: Argument classification results of the experiments with different annotations for arguments. The number above the bar shows the F_1 score for the UNIFIED arguments setting in the four data sets (IMS, J-A, J-C, J&L). The bars show the difference ΔF to each other setting.

real aspects, we hope that the drop is small. Indeed, micro-averaged F_1 score drops by only about 0.3 to 0.5 points. Recall that for the J&L and the JDPA data, the splitting of multiword predicates has been done automatically, so results are a bit noisy.

When we look at the confusion matrices for all data sets, we see that there are nearly no confusions of the scale with an entity. We have analyzed some cases where the scale has been confused with the aspect in the IMS data. Confusions occur mostly with untypical scale arguments like “*more feature rich*” or “*more pro*”, where the system predicts an aspect because the content word is tagged as a noun. We also found a few annotation errors where annotators mistakenly annotated an aspect instead of a scale.

For the last two experiments, we map the scale annotations back to aspects, but split the entities into two types: first (E1) and second (E2) entity for SURFACE entities, and preferred (E+) and non-preferred (E-) for PREFERENCE entities. The J&L data contains no information about the ranking of entities, so the PREFERENCE setting cannot be applied. Splitting the entities into two different types, no matter which, results in a drop in performance. As expected, it is much easier to distinguish entities by their position with regard to the predicate than by preference. But both settings have a larger influence on performance than the SCALE setting.

It is a bit surprising that the seemingly easy distinction of entities by position should make such a large difference in terms of performance (-4.2 points on the IMS data). From the confusion matrix it is clear that the difference actually comes from confusing the two entities, the number of wrong decisions between entities and aspects stays the same. There are two reasons for mistakes. One is, that the features for the classification include the position relative to the predicate, but they do not include information about

the other arguments. And while in most of the cases the first entity is before the predicate and the second entity after, this is not always the case. Additionally, even knowing about the other arguments would not completely solve the issue, as both entity roles can appear multiple times. Consider the following examples as illustration:

- (5.14) a. “**Like** [the SD550]_{E1}, [the SD800]_{E2} also have a 2.5 inch display at 7.1 megapixels.”
 b. “Whereas [the 630]_{E1} is a little **larger** than [the 710]_{E2} or [570]_{E2}.”

Determining which entity is preferred is more difficult than distinguishing two positions, and performance suffers even more (-8.8 points on the IMS data) Making the decision is often hard even for humans and sometimes impossible without more context or knowledge about the domain. Here are some hard examples:

- (5.15) a. “[The unit]_{E?} is slightly **larger** then [the D50]_{E?}.”
 b. “**Unlike** [the D200]_{E?}, [the D80]_{E?} can not [use non-CPU manual lenses]_A.”

To judge which entity is preferred, one needs to know whether being “large” or being able to “use non-CPU manual lenses” is something good or bad for the product. Whether “large” is good or bad does not only depend on the domain, but also on the target. A large memory is good for a camera, but a large file size is bad. The decision may even be subjective, some people like large cameras while others do not. Including this type of world knowledge into a system will be rather difficult if not impossible.

5.3.3. Discussion

The previous section presented experiments about the influence of different types of context on the performance of our comparison detection system CSRL. We have presented the detailed results of each experiment in the previous section, but during the analysis of the results we discovered some larger, more general issues which we discuss in this section. We focus on three topics: Linguistic anchoring of predicates and arguments, sentiment relevance and data sparsity.

Linguistic anchoring. The task of comparison detection in reviews is a relatively new task and many decisions about the exact formalization in terms of annotation guidelines and desired end results have yet to be taken. We have explored some of these decisions as they relate to multiword predicate in our experiments, but this is not the only difficult decision with regard to linguistic anchoring.

We have defined a comparison in our annotation scheme to have only one predicate, but there are cases where two words are used to express the same comparison. Consider these example annotations from the JDPA cameras data set:

- (5.16) a. “[**Lighter**]_A in weight compared to the [others]_E.”
 b. “. . . [its]_E [better]_A and faster **compared** vs the [SB800 flash]_E as well.”

While the sentences are similar in structure, the annotated predicates are very different. For our data, we have specified that the content word should be annotated, if possible, but this can still lead to situations where similar sentences may end up with different choices for the predicate. These are two examples of from our data where each annotation is correct and consistent on its own:

- (5.17) a. “Compared to [my previous powershot S50]_{E1} the [battery life]_A is much **better**.”
 b. “**Compared** to [my previous powershot S50]_{E1} the [battery life]_A is [good]_S.”

The anchoring of a comparison at a predicate is not the only issue, there is some variety of annotations on arguments as well. Part of the problem is the sometimes difficult distinction between entity and argument, which has been discussed already in the annotation guidelines for our data (see Section 4.5.3). On a more technical note, we introduce some variation by mapping all annotations to single words for our classification system. We chose the head of the subtree that contains the annotation for this purpose, which usually works well, but for variations of a long product name it can lead to different anchorings for very similar arguments. For example the product “*Canon Rebel XT*” appears in our data as “*the Canon Rebel XT model*” (head “*model*”), “*The new Rebel*” (head “*Rebel*”), “*the Canon Rebel XT*” (head “*XT*”) and more. In combination with spelling errors and applying a parser trained on news data to the noisy product reviews, there is a considerable variety of annotated tokens that confuse the system.

Another anchoring issue are implicit aspects, i.e., aspect contained in the predicate as exemplified in the following sentences:

- (5.18) a. “[it]_E is **close** in [weight]_A to [other cameras]_E of this type and quality.”
 b. “The magnesium alloy frame does add weight so [its]_E **heavier**.”

Both sentences compare the same attribute of the cameras (the weight) but only one of the sentences contains an overt realization of that aspect. In the other sentence, the aspect is implied in the predicate “*heavier*”. This is often the case for attributes like weight, speed, price or quality. In the JDPA data, the comparative adjective is often annotated as aspect in many such cases. We have decided not to duplicate annotations this way, as it leads to more confusing annotations in other cases. But this means that if the results of comparison detection are to be used for aspect-based summarization, some post-processing would need to be done to map implicit aspects.

	IMS	J-A	J-C	J&L
predicate annotations	2778	1327	642	698
distinct predicates	294	252	148	122
predicates with only one occurrence	146 (50%)	146 (58%)	89 (60%)	60 (49%)
predicates with ≥ 10 occurrences	48 (16%)	25 (10%)	12 (8%)	18 (15%)

Table 5.2.: Statistics about the number of predicate annotations in the data.

Sentiment Relevance. When we look at the false positives our system produces, we see a pattern that is familiar from other sentiment tasks where opinionated content has to be distinguished from non-opinionated context. In our case, often comparison words appear in non-comparative contexts, which are very hard to distinguish from comparative contexts. Consider the following example sentence:

(5.19) “*Relatively **lower** [noise]_A at higher ISO, but **higher** then [Sony]_E.*”

Although “*higher*” is often a predicate, and indeed is a predicate in the second part of the sentence, it is not always a predicate. In the example sentence, the phrase “*higher ISO*” only describes a camera setting and “*higher*” should not be marked as a comparative predicate. This type of usage is relatively common in our domain for descriptions of settings, features or environments. Applying techniques from subjectivity classification could help to find and exclude such occurrences (see also Section 2.2.1).

Data sparsity. There are many ways to express a comparison and the size of the available training data is relatively small. This strongly influences the recall of our system, as many predicates and arguments occur only once.

Table 5.2 shows some numbers, where we can see that about 50–60% of predicates occur only once, while only 8–16% occur ten times or more. Learning from such a sparse set is hard for any machine learning system. One reason for the large variety of different predicates is the widespread use of colloquial expressions in our user-generated data. This particularly affects verbs and nouns as predicates. These are some example sentences from the JDPA data:

- (5.20) a. “*In our store we routinely sold Nikon SLRs with [Tamron lenses]_E because they **hammer** the [Gs]_E.*”
 b. “*> [Nikon]_E **pwns** most other [companies]_E here .*”
 c. “*For users of prosumer cameras who want to upgrade to a higher hardware level , the [EOS 40D]_E **has the edge over** the [400D]_E if budget is not a concern.*”
 d. “*A quick search on Google will show the magnificent [Oly 50mm Macro]_E putting the equivalent [Leica]_E **to the sword** at a quarter the price.*”

For arguments, percentages of occurrences are similar. For entities, apart from the fact that different reviews discuss different models, a lot of the sparsity comes from the large variety of ways to refer to one and the same product, e.g., the camera “*Canon EOS 5D Digital SLR*” is referred to as “*EOS 5D*”, “*5D*”, “*Canon 5D*”, “*the Canon*”. Abstracting over the many model numbers is one way to go as seen in the improvement gained from NER. A large variety of different aspects are discussed in a lot of detail, because cameras are complex products. These aspects form a hierarchy and while Brown clusters can capture some of the similarities, the issue is more complex (see also Section 2.2.5).

One way to deal with sparse data is to enlarge size of the available training data. The most effective approach to do that is of course to invest in quality-controlled manual annotation of a relatively large amount of training data, however, this may not always be possible. The next chapter (Chapter 6) deals with the question of whether we can expand a small training set of labeled seed data by exploiting the structure of comparisons with a semi-supervised expansion algorithm.

5.4. Summary

In this chapter, we have presented work on comparisons, our second example of a complex verbalization of opinions. We introduced our system CSRL (Comparison Semantic Role Labeler) for the automatic detection of comparative predicates and their arguments. We have investigated the influence of different types of context information on our system. For argument identification and classification, we were able to validate our hypothesis that structural linguistic context from a dependency parse is more helpful than window-based context information. Different forms of task and domain context (Brown clusters, rule-based detection of product names, comparison type and predicate polarity) yielded mixed results. Finally, we have investigated the context created by decisions taken in annotation design and presented a detailed analysis of the different linguistic anchorings of multiword comparative predicates (“*more powerful*”, “*as good as*”, ...) and entities.

6. Semi-supervised training set expansion with structural alignment

The approach presented in this chapter and the test set experiments except those including predicate similarities have been published in (Kessler and Kuhn, 2015).

6.1. Introduction

In the two previous chapters we have discussed our corpus of annotated product comparisons (Chapter 4) and CSRL, our machine learning system for the automatic detection of comparisons and their components (Chapter 5). We have argued that comparisons are presumably the most useful kind of expression when it comes to supporting a process of choice: there tends to be a substantial proportion of reviews (about 5–10% of sentences) that include explicit textual comparisons, and the reviewer’s reason for the comparative assessment can be captured and fed into the choice process. To the extent that such subjective comparisons can be captured reliably by automatic means, they can provide an extremely helpful basis for coming up with a decision.

One insight from both, the annotations of our data and the training of our system, is that there is a large variety of comparison expressions and despite our data being the largest resource of comparisons currently available, sparsity is still an issue. Moreover, vocabulary differences across product categories make it advisable to use domain-specific training data. For example, while “*unpredictable*” is a positive description for the plot of a movie or book, it is negative when used to describe the steering behavior of a car. If enough (human and/or financial) resources are available, the most effective approach is of course to invest in quality-controlled manual annotation of a relatively large amount of training data, however, this may not always be possible. Fortunately, while there is a high variability in vocabulary of the actual textual realization of comparisons, the

higher-level structure of comparisons follows the pattern outlined in Chapter 4 independent of the domain: besides some comparative predicate that serves as indicator that a comparison is made, two entities are involved and usually one specific aspect of these entities is compared. While the actual vocabulary may differ, there are semantic constraints on the words that can be used as predicates, entities, and aspects and there is a limited number of valid syntactic variations. As an illustration, consider the similarities in the annotation of these examples taken from the data by Jindal and Liu (2006b):

- (6.1) a. “*first and foremost, [it]_{E1} will take **better** [pic’s]_A than [most film camera’s]_{E2}.*”
 b. “*as for [arsenal]_{E1}, i think they have a **better** [team]_A than [chelsea]_{E2}.*”
 c. “*Here is my feeling: [Pepsi]_{E1} is the **lighter** [taste]_A, where as [Coke]_{E2} . . .*”

Since the higher-level structure of comparisons as they appear in reviews is clear-cut, the problem setting could respond favorably to semi-supervised training strategies that start out from a small seed set of manually annotated data which is relatively cheap to create. As we have found the approach from Semantic Role Labeling (SRL) to be successful for the detection of comparisons, we look again to SRL for methods to expand a small seed set for our task. Several such methods have been proposed and successfully applied to SRL in the past years (Gildea and Jurafsky, 2002; Swier and Stevenson, 2004; Fürstenaу and Lapata, 2009; Franco-Peña and Emms, 2012).

The method we have selected is structural alignment, proposed by Fürstenaу and Lapata (2009, 2012). This approach covers all steps of the SRL pipeline and does not depend on SRL-specific resources or corpora. Structural alignment is a label projection algorithm. Starting from a small set of labeled seed sentences, the algorithm attempts to find sentences that are similar to these seed sentences. Similarity is compared based on the syntactic and semantic contexts of the predicates. The basic hypothesis is that predicates that appear in a similar syntactic and semantic context will behave similarly with respect to their arguments. For the most similar sentences that have been found in this way, the labels from the seed sentences are projected onto the unlabeled sentences. The newly annotated sentences are then added to the training data for the original task, in our case comparison detection. In this way, a small manually annotated seed set of sentences can be expanded to create a set large enough for efficient machine learning without additional manual annotation effort.

There are several challenges that make our task different from the typical setting in which SRL operates. First, our data is not news text, but user-generated data (product reviews), which is much more noisy, containing unknown words as well as errors in grammar, spelling and punctuation. This contributes to the sparsity of our data, as well to a larger amount of unreliability in the NLP tools we are using, such as part of speech

tagging or parsing. Second, we have a smaller, more fixed set of roles for the arguments than in traditional SRL (two entities that are compared in one aspect), but these arguments are generally further away from the predicates and there is a greater variety of syntactic expressions. This makes it necessary to extend the search space for argument candidates and to include more context into our similarity measures. Finally, like all sentiment-related task, we have to deal with sentiment relevance (cf. Section 2.2.1). Not every occurrence of a word that looks like a comparative predicate actually introduces a comparison.

In the following, we investigate whether, despite these differences, structural alignment can be used successfully for getting additional training data for the task of comparison detection. Capturing the structure of comparisons is the key step in this process to ensure that the found sentences are similar enough so that the projection of labels onto them produces correctly labeled new training examples, but also different enough to include linguistic variations that enable the classifier to learn new information. We present some adaptations of structural alignment to our task, tailoring the approach to the structural characteristics of comparisons. To evaluate our approach, we directly determine whether the projected labels correspond to manually assigned labels in a small development set, and we also compare the performance of CSRL when trained on the original seed data and when trained on the expanded data set, experimenting with varying numbers of seed sentences and gathered expansion sentences.

In summary, this chapter deals with research question B:

Research Question B: *How can the structure of complex verbalizations of opinions be exploited in order to automatically annotate training examples by using semi-supervised methods?*

More specifically, we explore the following questions:

- Can structural alignment, a semi-supervised method that has been successfully used for projecting SRL annotations to unlabeled sentences, be adapted to the task of detecting comparisons?
- How can we adapt the argument candidate creation step to reflect that our arguments are further away from the predicate, while at the same time keeping the number of candidates manageable?
- What is the best way to include sentence context into the similarity measures for arguments to improve the created alignments?
- Does including information about the predicate into the alignment similarity measure help to filter out false positive matches?

6.2. Semi-supervised learning and related work

In *supervised machine learning* (classification), we assume the existence of some training data, a set of examples X with assigned classes or labels Y , from which we can learn a function $f : X \rightarrow Y$ (a model) that maps the examples to the correct labels. Conversely, in *unsupervised machine learning*, we do not have label information Y for the examples. Instead of learning a mapping from X to Y , we can only group the examples into categories according to the internal structure of the data. *Semi-supervised* or weakly supervised learning lies in between supervised and unsupervised learning. In this setting, while we have some information about the labels on the training data, other information may be missing. Semi-supervised learning attempts to leverage unlabeled data in combination with the labeled data to find the missing information. The basic case of semi-supervised learning has labels for only part of the training data, but there are other forms of semi-supervised learning that for example only have partial or noisy label information or use a set of constraints as basis for labels. For a discussion of semi-supervised learning in computational linguistics see Abney (2007).

A frequently used method of semi-supervised learning is *self-training* or bootstrapping. Typically, self-training starts with a small set of labeled training examples on which a supervised classifier (the *base learner*) is trained. This classifier is then applied to a larger set of unlabeled examples and assigns a label and a confidence score to each example. The examples where the classifier is most confident are then added to the training data. A new classifier is trained on the expanded training data, applied to the unlabeled data and again the most confident examples are added to the training data. The process is iterated until some stopping criterion is reached, for example be a fixed number of iterations or until convergence, i.e., the classifier or the labels of the data do not change from one iteration to the next. One well-known example of a bootstrapping approach used in NLP is the Yarowsky algorithm (Yarowsky, 1995). There are many variants of this basic method depending on the exact implementation of each step.

Semi-supervised approaches have been used in many fields and Semantic Role Labeling is no exception. Already Gildea and Jurafsky (2002), the first work to tackle SRL as an independent task, use bootstrapping to enlarge their training data. Their final expanded training data is about six times the size of the originally annotated training data and they report a small improvement for training their system on the expanded versus training on the non-expanded training data.

Other approaches use the extensive resources with information about predicates and possible arguments that exist for SRL as a basis for bootstrapping. The work of Swier

and Stevenson (2004, 2005) leverages VerbNet as the basis for a bootstrapping approach to classify argument roles. VerbNet lists possible argument structures allowable for each predicate. For a given argument, they determine the set of possible roles from VerbNet. After initially making all unambiguous role assignments, their system learns from these assignments and iteratively proceeds to label all arguments.

Apart from bootstrapping, *label projection* is another common method to assign labels to unlabeled data. Label projection starts with an individual labeled seed example, looks for similar examples, and then projects the labels over some alignment between the two examples. Label projection has been used in multilingual contexts for various tasks, from projecting parts-of-speech and chunks (Yarowsky et al., 2001) to Named Entities (Ehrmann et al., 2011) and also to project SRL information from one language to another (Padó and Lapata, 2009). In multilingual contexts, the alignments that are used for the projection are derived from the translation alignment between the words of the original sentence and its translation. For monolingual contexts a different sort of alignment with different measures of similarity is necessary. Such monolingual alignments are studied for in textual entailment recognition or paraphrase identification, although for these tasks there is usually no transfer of labels along the alignments (Yao et al., 2013).

The approach we are adopting for our work in this chapter is structural alignment proposed by Fürstenaу and Lapata (2009, 2012). While they also use a bootstrapping approach in their evaluation, their proposed method is a label projection algorithm over sentences. The process is based on the assumption that sentences which are similar in syntactic structure and semantics of the arguments also have similar predicate-argument relations. For each labeled seed sentence, the k most similar sentences are extracted from a large set of unlabeled sentences and the labels of the seed sentence are projected. Structural alignment is explained in detail in the following section together with our adaptations. Fürstenaу and Lapata (2009, 2012) address the complete pipeline of SRL steps, from predicate identification until argument classification.

A similar approach is presented also by Franco-Peña and Emms (2012), but their approach relies on already identified predicates and arguments and is thus not applicable to our task, where we do not have such information. For an unlabeled sentence, they project the labels for all arguments from the labeled sentence with the smallest tree edit distance and experiment with different cost measures. In contrast to the above approaches, they do not use their method to expand a training set for classification, but directly classify the unlabeled sentences (similar to 1-Nearest-Neighbor classification).

Some completely unsupervised approaches have also been proposed for tasks in the SRL pipeline. Abend et al. (2009) do unsupervised argument identification by using

pointwise mutual information to determine which constituents are the most probable arguments. Initially, all constituents are regarded as argument candidates, this set is then filtered (by using minimal clauses, pruning and pointwise mutual information) to include only the most probable candidates. Their work is limited to argument identification and they use gold predicates as a starting point. In contrast to their work, we cannot start from given predicates in the test set, as we have no annotations at all for our unlabeled sentences.

For comparison detection we do not have extensive resources like PropBank or VerbNet at our disposal. We do however think that a small seed set of comparison sentences can be annotated in reasonable time for any new domain or language. This set may not be sufficiently large for bootstrapping, but it can be used as an initial seed set for a label projection approach like structural alignment.

6.3. Structural alignment approach

The goal of our work is to get more training data for our task of comparison detection by expanding a small set of labeled seed sentences. We have implemented structural alignment proposed by Fürstenaу and Lapata (2009, 2012), a semi-supervised label projection method for finding unlabeled sentences that are similar to labeled seed sentences. The basic hypothesis is that predicates that appear in a similar syntactic and semantic context will behave similarly with respect to their arguments. Based on this, the labels from the seed sentences can be projected to the unlabeled sentences. The newly labeled sentences can then be used as additional training data for the original task.

We first give the general outline of structural alignment in the next subsection and then give more specific details for each of the separate steps in the later subsections. Our adaptations to the new task of comparison detection mainly focus on argument candidate creation and alignment scoring, as we assume these two steps to have the biggest effect on overall performance. For the other steps, we have made only minor changes to the original approach.

6.3.1. Outline of structural alignment

The pseudo-code for structural alignment is given in Figure 6.1. This section presents a rough outline of the process of label projection with references to the following sections where we discuss each step of the expansion algorithm in detail. Figures 6.2 and 6.3 illustrate each step for a pair of example sentences from our data.

```

1: for seed sentence  $s \in S$  do
2:   for predicate  $p \in s$  do
3:      $L \leftarrow \emptyset$  ▷ list of expansion sentences
4:     for  $u \in U$  do ▷ process all unlabeled sentences
5:        $\sigma(p) \leftarrow \text{getCompatiblePredicate}(p, u)$ 
6:       if  $\sigma(p) \neq \epsilon$  then
7:          $M \leftarrow \text{getCandidatesLabeled}(s)$ 
8:          $N \leftarrow \text{getCandidatesUnlabeled}(u)$ 
9:          $(\sigma, \text{score}_s) \leftarrow \text{getBestAlignment}(M, N)$ 
10:        if at least one role-bearing node is covered in  $\sigma$  then
11:          add  $(\sigma, \text{score}_s, u)$  to  $L$ 
12:        end if
13:      end if
14:    end for
15:    sort  $L$  by descending alignment similarity scores
16:    for  $i = 1$  to  $k$  do ▷ project labels for the best  $k$  sentences
17:      projectLabels( $s, L(i)$ )
18:    end for
19:  end for
20: end for

```

Figure 6.1.: Outline of structural alignment in pseudo-code.

We are provided with a small set of labeled sentences S (seed data) and a large set of unlabeled sentences U (expansion data). We collect expansion sentences individually for every predicate of every labeled sentence from the seed corpus. If a sentence has two or more predicates, the process is repeated for each predicate independent of the others. As compatible predicates and extracted argument candidates will differ for different predicates, the final expansion sentences will usually not be the same for predicates from the same sentence.

We start with a given predicate p of seed sentence s , for example the predicate “*higher*” from the following sentence (the second predicate of the sentence, “*same*”, would be considered in a separate step):

(6.2) “[*This camera*]_{E1} has just a bit **higher** [*learning curve*]_A than [*the Canon SLRs I’ve used*]_{E2} but about the same with the Nikons.”

The following steps are performed for every unlabeled sentence $u \in U$. First, we check whether u contains a predicate which is compatible with p (cf. Section 6.3.2). We denote the predicate candidate in u with $\sigma(p)$, the word in the unlabeled sentence which is aligned with p . We filter by predicate, because the possible variations of comparison expressions are depending on the predicate, so similar sentences will always have the

same or at least a very similar predicate. If we take compatible predicate to mean a word with the same part of speech, we might find the following unlabeled sentence for our example seed sentence where $\sigma(p)$ has been marked in bold:

(6.3) “The camera has a somewhat **larger** body than many digital cameras but it’s still easy to hold onto.”

For sentences where there is a compatible predicate, we proceed to get all argument candidates M from the labeled seed sentence s and all argument candidates N from the unlabeled sentence u (cf. Section 6.3.3). The method of obtaining the candidate sets may be different for the labeled and the unlabeled sentence, e.g., we can take the actual arguments on labeled side and syntactically related word of $\sigma(p)$ on unlabeled side. From the two example sentences, we could for example extract the following candidates:

- Candidates M from s : “camera”, “curve”, “SLRs”
- Candidates N from u : “camera”, “has”, “a”, “somewhat”, “body”, “cameras”

We then score every possible alignment between the two argument candidate sets M and N , to find the best possible correspondence between the argument candidates in the two sentences. Scoring is done by summing up the similarities of all aligned words (cf. Section 6.3.4–6.3.6). We select the best-scoring alignment σ for sentence u and store it together with the corresponding score $score_s(s, u)$ iff at least one role-bearing node is covered. For our example sentence u we store the following best alignment:

- Alignment score: $score_s(s, u) = 0.75$
- Alignment: $\sigma(\text{camera}) = \text{camera}$, $\sigma(\text{curve}) = \text{body}$, $\sigma(\text{SLRs}) = \text{cameras}$

When all unlabeled sentences have been processed, we choose the k sentences with the highest alignment similarity scores as expansion sentences for the seed predicate p of sentence s . These are the sentences with the most similar arguments, so they presumably can be labeled in the same way as our original sentence. For example, from the following three sentences with their scores, for $k = 1$ we chose the only the first sentence:

- (6.4) a. “The camera has a somewhat **larger** body than many digital cameras but it’s still easy to hold onto.” ($score_s(s, u) = 0.75$)
- b. “DSC-H2 seemed to drain batteries at a **higher** rate than what specifications stated it would.” ($score_s(s, u) = 0.54$)
- c. “You should stick to ISO 50, there is no need for **higher** ISO or using flash (unless pitch black) as long as you turn on stabilization.” ($score_s(s, u) = 0.39$)

For the chosen sentences, we project the labels of the arguments in the seed sentence onto their aligned words in the unlabeled sentence (cf. Section 6.3.7) and add the newly labeled sentences to the training data. Our example processing results in this:

(6.5) “The [camera]_{E1} has a somewhat **larger** [body]_A than many digital [cameras]_{E2} but it’s still easy to hold onto.”

The following sections talk in more detail about our specific processing for sentence selection (Section 6.3.2), argument candidate creation (Section 6.3.3), similarity scoring for arguments (Section 6.3.4) and predicates (Section 6.3.5), word alignments (Section 6.3.6), and label projection (Section 6.3.7).

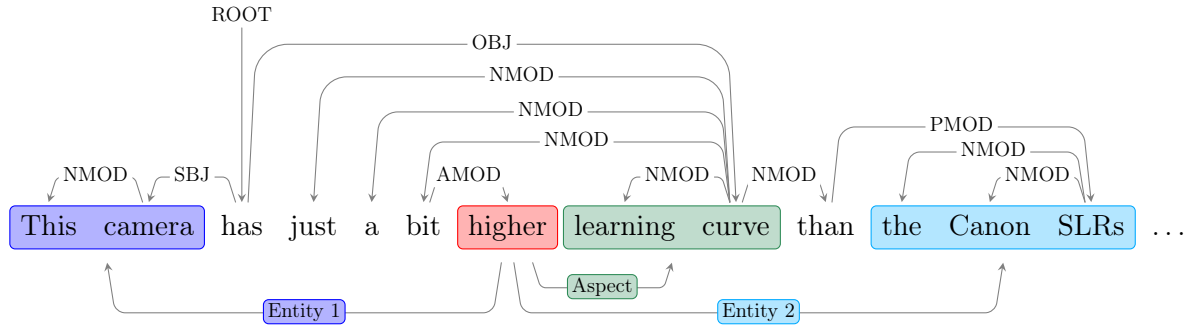
6.3.2. Sentence selection

Generally, following Fürstenaу and Lapata (2009, 2012), we consider all unlabeled sentences to be possible expansion sentences that contain a word with the exact same lemma as the labeled predicate. Additionally, in contrast to the original approach, we use the part of speech (POS) tag instead of the lemma for all adjectives and adverbs in comparative or superlative form (see the example in Figure 6.2), as exchanging the exact word in this case is without any influence on the syntactic structure or the arguments of the comparison. Like the original approach, we only consider single-word predicates.

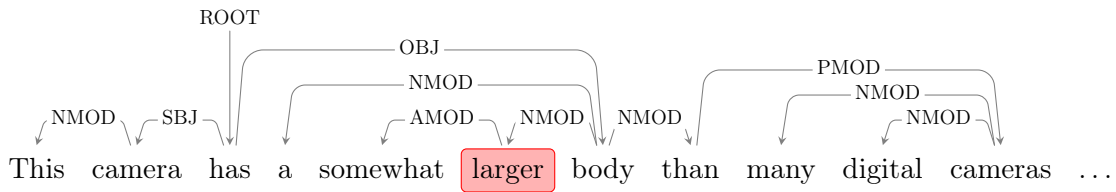
6.3.3. Creation of argument candidates

The basis for extracting argument candidates is the dependency parse of the sentence. A dependency parse models the relations between pairs of words in the sentence. Figure 6.2 shows the dependency parse trees for an example seed sentence and an example unlabeled sentence. Each word is a node in the tree. There is one *root* node, usually the main verb. In the example it is the word “has” in both sentences. A relation between two words is called a *dependency* and has a type that gives information about the grammatical function, e.g., SUBJ for subject, AMOD for adjectival modifier. The dependency is directed from the *dependent* (the governed word, the child node) to the *head* (the governing word, the parent node).

For the extraction of argument candidates, the anchor in the sentence is the predicate and we use the following terms and notation to describe the relationship of a candidate with the predicate. We refer to the dependents of the predicate and all their dependents as *descendants*, i.e., this includes all nodes that are reached when going “down” in the tree from the predicate (symbolized by ↓). Similarly, *ancestors* refers to all words reached when going “up” in the parse tree from the predicate until the root node (symbolized by ↑). When a word can be reached by taking only one step “up” respectively “down”, we call it a *direct descendant*, respectively the *direct ancestor*. When a word is a direct



(a) Labeled seed sentence with predicate “higher/JJR”.



(b) Unlabeled expansion sentence with compatible predicate “larger/JJR”.

LABELED		“camera”, “curve”, “SLRs”
FL	Descendants:	–
	for “camera”:	“bit” (↑), “curve” (↑↑), “has” (↑↑↑), “camera” (↑↑↑↓)
	for “curve”:	“bit” (↑), “curve” (↑↑)
	for “SRLs”:	“bit” (↑), “curve” (↑↑), “SRLs” (↑↑↓)
KK	Descendants:	–
	Ancestors:	“bit” (↑), “curve” (↑↑), “has” (↑↑↑)
	Siblings of “bit”:	“just” (↑↑↓), “a” (↑↑↓), “learning” (↑↑↓)
	Siblings of “curve”:	“SRLs” (↑↑↓, prep. collapsed), “camera” (↑↑↑↓)

(c) Argument candidates on labeled side with different methods (LABELED, FL, KK).

FL	Descendants:	“somewhat” (↓)
	for “camera”:	path (↑AMOD ↑NMOD ↑OBJ ↓SBJ) not found
	for “curve”:	path (↑AMOD ↑NMOD) not found
	for “SRLs”:	path (↑AMOD ↑NMOD ↓NMOD ↓PMOD) not found
KK	Descendants:	“somewhat” (↓)
	Ancestors:	“body” (↑), “has” (↑↑)
	Siblings of “larger”:	“a” (↑↓), “cameras” (↑↓, prep. collapsed)
	Siblings of “body”:	“camera” (↑↑↓)

(d) Argument candidates on unlabeled side with different methods (FL, KK).

Figure 6.2.: Argument candidate extraction with different methods for an example labeled seed sentence and an unlabeled expansion sentence.

descendant of the same word as the predicate, i.e., reached by going one word “up” and one word “down” (symbolized by $\uparrow\downarrow$), we call it a *direct sibling*. Similarly, *siblings* in general are all descendants of direct siblings.

Fürstenau-Lapata path-based method (FL). Fürstenau and Lapata (2009) use all direct descendants and direct siblings of the predicate as argument candidates (on labeled side this includes both SRL arguments and non-arguments). Their main assumption is that “we expect all roles in the graph to be instantiated by syntactic dependents of [the predicate] p ” (Fürstenau and Lapata, 2009). When we extract direct descendants and direct siblings of all predicates in our labeled data, we find that only 17% of the actual labeled arguments are contained in this set (5% of arguments are direct children, 12% direct siblings of a predicate). For example, for the labeled sentence in Figure 6.2 the candidate set created with this method would not include any argument of the predicate, as no direct descendants and direct siblings of the predicate exist.

In their follow-up work (Fürstenau and Lapata, 2012), the authors additionally extract the dependency paths from the predicate to each argument in the labeled sentence. Paths include all the dependency relations between the predicate and the argument and their direction in the tree (\uparrow or \downarrow). For the labeled seed sentence in Figure 6.2, we would for example extract the path (\uparrow AMOD \uparrow NMOD \uparrow OBJ \downarrow SBJ) for the argument “*This camera*”. They search for these exact same paths in the unlabeled sentence, and extract all nodes on these paths as argument candidates. We use **FL** to refer to this Fürstenau and Lapata’s method which includes the **direct descendants and the exact paths to the arguments** in the argument candidate set.

In contrast to the news data Fürstenau and Lapata (2009, 2012) work on, we have to deal with noisy user-generated data, where parses are less reliable (cf. the example seed sentence, where “*a bit*” should modify “*higher*”, instead “*higher*” is wrongly attached to “*bit*”). Additionally, comparative arguments tend to be further away from their predicates than SRL arguments, e.g., often one of the entities is the subject in a sentence. As a result, the paths to the labeled arguments are diverse and sometimes very long. When we extract all paths from predicates to arguments from our labeled data, many paths are found only once, so we expect this method to fail often in finding any candidates. As a case in point, in the example unlabeled sentence, none of the paths extracted from the labeled sentence is found, so no candidates can be extracted from this sentence (there are of course other sentences in the data where the paths will be found). On the other hand, when an exact path is found, we expect the match to be very precise and give a well-matching candidate for the argument.

Kessler-Kuhn dependency-based method (KK). The challenge is to enlarge the set of argument candidates in order to increase the chances of finding correct alignments, while keeping the number of candidates manageable so that alignments can be calculated in reasonable time. In keeping with the original idea, we extract candidates based on their dependency relations with the predicate. But instead of using only the direct descendants, ancestors and siblings, we experiment with adding to the candidate set all descendants, all ancestors up until the root of the dependency tree, and all descendants of the siblings. For the labeled example sentence, this approach would add the actual argument “*curve*” to the candidate set (parent of the parent of the predicate, $\uparrow\uparrow$). If we look at statistics of the labeled data, 22% of arguments are children of a predicate (only 5% are direct children), 24% are ancestors (18% direct ancestors), and 15% are siblings (12% direct siblings). Adding all of these, we find 61% of real candidates, and the extracted sets contain between 1 and 25 items with an average size of 7.

Still, nearly 40% of candidates are not included when we extract candidates based on the above dependency relations. Lluís et al. (2013) observe that nearly all of their SRL arguments are either direct descendants of the predicate or direct descendants of an ancestor of the predicate. Accordingly, our method which we call **KK** (referring to Kessler and Kuhn) uses **all ancestors of the predicate until the root and their direct descendants, plus all descendants of the predicate itself**. With this, the candidate set for the example labeled sentence includes the first entity, “*camera*”, in addition to the still-included aspect “*curve*”. In the complete labeled set, we find about 86% of all arguments with an average candidate set size of 10.

Filtering and (a)symmetric extraction. To reduce the number of candidates that are very unlikely to be arguments, we impose a distance limit for candidates and exclude numbers and punctuation. We also remove prepositions (Fürstenau and Lapata, 2009) and conjunctions (Fürstenau and Lapata, 2012), as they can never be arguments themselves. We follow Fürstenau and Lapata (2012) and add their direct children to the candidate set. In the example labeled sentence, this would remove the preposition “*than*” and instead add the direct child “*SLRs*”, which finally leads to the example candidate set containing all actual arguments.

Fürstenau and Lapata (2012) include the predicate itself in the candidate set. As the predicate is the pivot for selecting the unlabeled sentence and will always be aligned with the word found there, we do not include these words into the candidate set, instead we consider predicates separately for alignment similarity calculation (see Section 6.3.5).

The original approach extracts candidates in a symmetric way, i.e., using the same

candidate extraction method on both the labeled and the unlabeled sentence. This means that the candidate set for the labeled side includes non-arguments as well. As our interest is solely in finding good alignments for the projection of the real arguments, and our candidate sets are relatively noisy, we use an asymmetric method for candidate extraction. On the labeled side, we use only the actual labeled arguments of the comparison. On the unlabeled side, we use the set extracted with the methods described above. We refer to the symmetric candidate extraction by FL-SYMM and KK-SYMM and the asymmetric candidate extraction by FL and KK.¹ Figure 6.2 shows the candidate sets for the labeled and unlabeled example sentences extracted with all methods described in this section.

6.3.4. Argument similarity scores

The similarity of an alignment between two sentences s and u is the averaged sum of all word alignment similarities. Word alignment similarities are themselves the averaged sum of different word similarity measures:

$$\text{score}_s(s, u) = \frac{1}{|M|} \sum_{i=1}^{|M|} \frac{1}{|J|} \sum_{j \in J} \alpha_j \cdot \text{sim}_j(w_i, \sigma(w_i)) \quad (6.1)$$

where M is the set of candidates on labeled side, $w_i \in M$ one of these candidates, $\sigma(w_i)$ the candidate on unlabeled side aligned with w_i , and J is the set of word similarities to calculate. The parameters α_j regulate the relative importance of each word similarity measure. Unaligned w_i receive a word similarity of zero.

This definition of alignment similarity follows the original approach proposed for SRL, but adds the possibility of including arbitrary similarity measures, each with an individual weight². The original approach uses only the weighted sum of syntactic (syn) and lexical/semantic (sem) similarity as word similarity measure. Choosing these two similarity measures follows the intuition that both complement each other and that sentences need to have both a similar syntactic structure and semantically similar arguments in order to be able to project the labels. Fürstenau and Lapata (2009) give this formulation

¹In (Kessler and Kuhn, 2015) the terms PATH-FILTERED and DEPENDENCY-FILTERED candidate creation are used, but we decided to switch terminology to avoid the confusion of these names with the PATH and DEPENDENCY similarity measures discussed in the following section.

²There are countless possibilities for word similarity measures, we discuss those that we use later in this section and test their effects on the development set in the experiments.

for their sentence similarity measure (adapted to our naming conventions):

$$\text{score}_{\text{FL09}}(s, u) = \sum_{i=1}^{|M|} (\alpha \cdot \text{syn}(w_i, \sigma(w_i)) + \text{sem}(w_i, \sigma(w_i)) - B) \quad (6.2)$$

which compared to Equation 6.1 contains the additional parameter B . This parameter is needed to allow for unaligned words in their ILP setup. Unaligned nodes are covered in our setup by setting their similarity value to zero. Fürstenau and Lapata (2009) do not normalize by number of arguments a predicate has. This is not necessary in their setup as similarity values are only compared with scores for the same seed sentence which has always the same number of arguments.

Fürstenau and Lapata (2012) use a formulation over dependency edges:

$$\text{score}_{\text{FL12}}(s, u) = \frac{1}{C} \left(\alpha \cdot \sum_{\substack{(w_i, x_2) \in E(M) \\ (\sigma(w_i), \sigma(x_2)) \in E(N)}} \text{syn}(w_i, \sigma(w_i)) + \sum_{\substack{w_i \in M \\ \sigma(w_i) \neq \epsilon}} \text{sem}(w_i, \sigma(w_i)) \right) \quad (6.3)$$

where $E(M)$ and $E(N)$ are the set of edges between nodes in labeled-side candidates M and unlabeled-side candidates N respectively. Now, C is introduced as a normalization factor to make scores comparable across sentences. They have also changed their setup to remove the need for the parameter B .

Semantic similarity is calculated as the cosine similarity of the co-occurrence vectors of the two words, co-occurrences are extracted from news texts. We use the same measure as our **vector space similarity** (Vs) with context vectors extracted on texts from our domain. As a measure for syntactic similarity, they compare the dependency relation of the candidates (1 if they are the same, 0 if they are different). For our **flat syntactic similarity** (DEP) we take into account that our sentences are more noisy and parses are not as reliable. To the comparison of dependency relations (0.5 same, 0 different) we add a comparison of the parts of speech, as these are more reliable (0.5 same, 0.25 same universal part-of-speech tag, 0 different).

A similarity measure that looks only at words without context works in a situation where there are relatively few candidates to chose from. But with a larger and noisier pool of candidates, there are too many similar candidates and the best match cannot be found without taking into account sentence structure and context. We have experimented with several possibilities to replace the flat syntactic similarity that only looks at the current candidate with a more context-aware similarity measure.

From the surface form of the sentence we can get a “flat” context for the argument

candidates. **Window similarity** (WINDOW) averages over the vector space similarity VS of the neighboring words to the left and right of a given argument candidate w_i :

$$\text{sim}_{\text{WINDOW}}(w_i, \sigma(w_i)) = \frac{1}{2}(\text{sim}_{\text{VS}}(w_{i-1}, \sigma(w_{i-1})) + \text{sim}_{\text{VS}}(w_{i+1}, \sigma(w_{i+1}))) \quad (6.4)$$

We can also compare the **token position similarity** (POSITION) of the argument candidate with respect to the predicate p . Specifically, we calculate the distance between argument and predicate in number of tokens d_{tok} , and compare the two distances:

$$\text{sim}_{\text{POSITION}}(w_i, \sigma(w_i)) = \frac{1}{|d_{\text{tok}}(w_i, p) - d_{\text{tok}}(\sigma(w_i), \sigma(p))| + 1} \quad (6.5)$$

WINDOW and POSITION are easy to implement as they work on the surface level, but to get more accurate information about the sentence structure, we can transform them to include information from the dependency parse tree of the sentence. Instead of “flat” window context for the candidate which uses the words directly besides the candidate, we can use the window context around the subtree that includes all children of the current candidate. As an example, take the candidate “camera” in the phrase “*bigger than my old camera*”, WINDOW would use “old” as context, while subtree neighbors would use “than”, because the other two words are dependents of the candidate. **Subtree window similarity** (TREETWINDOW) thus compares the vector space similarity VS of the word to the left of the leftmost token in the subtree and the word to the right of the rightmost token in the subtree below the argument candidate.

When comparing position in the parse tree, as a basic measure we can compare **level similarity** (LEVEL), i.e., the number of “up”s (d_{\uparrow}) and “down”s (d_{\downarrow}) on the dependency path from argument to predicate. Roughly this corresponds to comparing the difference in level in the parse tree relative to the predicate and is achieved by replacing the token distance from TOKENS with the “up” and “down” distance:

$$\text{sim}_{\text{LEVEL}}(w_i, \sigma(w_i)) = \frac{1}{2} \sum_{l \in \{\uparrow, \downarrow\}} \frac{1}{|d_l(w_i, p) - d_l(\sigma(w_i), \sigma(p))| + 1} \quad (6.6)$$

The full **path similarity** (PATH) measures the similarity of the words on the dependency path from argument candidate to the predicate. For each pair of words, the word similarity is calculated and the final path similarity is the average over all word similarities. We have experimented with different measures of similarity for the words on the path and found DEP to perform best. The “up” and “down” parts are counted separately and if paths have different length, every left-over word is added with similarity 0.

	name	level	arg	pred	compares ... of the two words
Sem.	VS	flat	✓	✓	cosine similarity
	WINDOW	surface	✓	✓	VS of left/right neighbors
	TREEWINDOW	dependency	✓	✓	VS of left/right neighbors of subtree
Syntax	DEP	flat	✓	✓	dependency relation and POS
	POSITION	surface	✓	–	token distance to predicate
	LEVEL	dependency	✓	–	ups and downs on dependency path to predicate
	PATH	dependency	✓	–	DEP of all words on dependency path to predicate

Table 6.1.: Overview of syntactic and semantic word similarity measures that can be applied to arguments (arg) and predicates (pred).

Table 6.1 contains an overview of all word similarity measures. In summary, we have two flat similarity measures (VS, DEP), two measures that include surface sentence context (WINDOW, POSITION) and three measures that include dependency parse context (TREEWINDOW, LEVEL, PATH). We test the effects of all these similarity measures on expansion accuracy in our experiments.

6.3.5. Predicate similarity scores

The originally proposed alignment similarity discussed in the previous chapter depends only on the arguments and does not directly consider the predicate itself. In SRL, every occurrence of a predicate word also functions as a predicate, the only question is whether it is an instance of the frame we are currently interested in or a different one. In our case though, we have many instances where occurrences of words that look like comparative predicates do not introduce a comparison. To this end, we propose to add a predicate similarity score to the alignment score, where information about the predicate candidate can be considered. We add the predicate similarity with the same weight as one argument, i.e., the alignment similarity between s and u now becomes:

$$\text{score}_s(s, u) = \frac{1}{|M| + 1} \left(\sum_{i=1}^{|M|} \frac{1}{|J_a|} \sum_{j \in J_a} \alpha_j \cdot \text{sim}_j(w_i, \sigma(w_i)) + \frac{1}{|J_p|} \sum_{j \in J_p} \beta_j \cdot \text{sim}_j(p, \sigma(p)) \right) \quad (6.7)$$

where J_a and J_p are the sets of similarities used for argument and predicate similarities respectively. Similar to α_j for the arguments, the parameters β_j regulate the relative importance of each predicate similarity measure.

Those argument similarity measures that are not dependent on the predicate can directly be used to measure similarity of predicates (VS, WINDOW, TREEWINDOW,

DEP). A ✓ symbol in the column ‘pred’ in Table 6.1 indicates that we use the measure for predicates. We would expect the context similarities WINDOW and TREEWINDOW to be especially useful.

6.3.6. Alignments between candidate sets

Input to the alignment step are two sets of argument candidates: the set on labeled side M with m elements and the set on unlabeled side N with n elements. Our goal is to find for each argument candidate w_i on labeled side one aligned candidate $\sigma(w_i)$ so that the overall alignment similarity is maximized. Any labeled candidate w_i can also stay unaligned, i.e., aligned to nothing. Unlabeled candidates can also stay unaligned. This will occur relatively frequently if the source side set contains only the real arguments (between 1 and 4 usually), as there will usually be more candidates on unlabeled side.

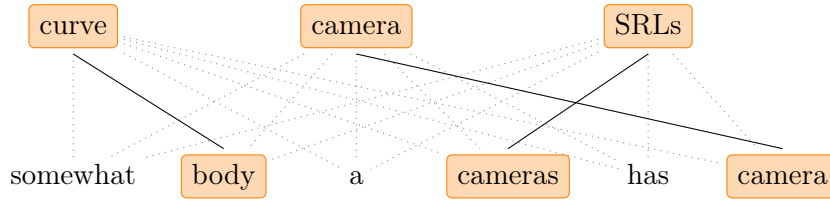
This setting is an instance of the assignment problem, originally formulated by Koopmans and Beckmann (1957) as finding the best assignment of m workers to n tasks, where each worker has a rating for the tasks that indicates some measure of cost. In our case we need to find an assignment between the two sets of candidates given a measure of similarity between every pair of words. More formally, the assignment problem can be formulated as follows. Let G be a (complete) weighted bipartite graph with $V = M \cup N$ with $M \cap N = \emptyset$ and $E \subseteq M \times N$. Every edge $v \in E$ is assigned a weight which represents the “cost” of assigning the worker m to the task n . A matching is a subset $A \subseteq E$ such that $\forall v \in V$ at most one edge in A is incident upon v (i.e., every node has at most one edge that connects it to the other partition of the graph). The assignment problem is to find a minimum weight matching in G .

We use the Hungarian Algorithm (Kuhn, 1955) to solve the problem which has a time complexity of $\mathcal{O}(n^3)$. To apply the Hungarian Algorithm, we calculate the word-similarity for every pair of words $(w_i, \sigma(w_i))$ from our two candidate sets where $w_i \in M$ and $\sigma(w_i) \in N$. This results in a $m \times n$ matrix with similarity values in the range of $[0, 1]$. The Hungarian algorithm works to find minimum costs, not maximum similarity, so we use $1 - \text{similarity}$ as entries for the final matrix.

The Hungarian Algorithm now performs the following steps on the input matrix³:

1. Subtract the smallest entry in each row from all the entries of its row.
2. Subtract the smallest entry in each column from all the entries of its column.
3. Cover all zeroes with the smallest possible amount of vertical and horizontal lines, i.e., lines that cover either one row or one column of the matrix.

³See a step-by-step explanation and solve cost matrices at <http://www.hungarianalgorithm.com>.



(a) Argument candidates on labeled side (upper line, real arguments) and unlabeled side (lower line, extracted with KK) with all possible alignments between them (dotted lines) and one selected alignment (solid lines).

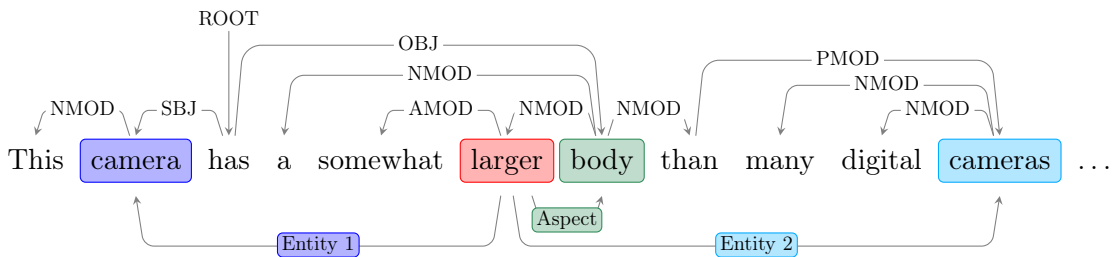
sim_j	value w_i	value $\sigma(w_i)$	$sim_j(w_i, \sigma(w_i))$
VS	$\vec{v}(slrs)$	$\vec{v}(cameras)$	0.91
WINDOW	$\vec{v}(canon), \vec{v}(i)$	$\vec{v}(digital), \vec{v}(but)$	0.61
TREEWINDOW	$\vec{v}(than), \vec{v}(i)$	$\vec{v}(than), \vec{v}(but)$	0.78
DEP	PMOD, NNP	PMOD, NNS	0.75
POSITION	6	5	0.50
LEVEL	$\uparrow 2 \downarrow 2$	$\uparrow 1 \downarrow 2$	0.75
PATH	\uparrow bit \downarrow curve, than	\downarrow body, than	0.70
$score_w("SLRs", "cameras")$			0.71

(b) Word similarities for "SLRs" as w_i and "cameras" as $\sigma(w_i)$.

	"camera"	"has"	"a"	"somewhat"	"body"	"cameras"
"camera"	0.82	0.38	0.32	0.34	0.44	0.34
"curve"	0.46	0.33	0.31	0.24	0.70	0.35
"SLRs"	0.38	0.28	0.31	0.25	0.37	0.71

$$score_s(s, u) = 1/3 \cdot (0.82 + 0.70 + 0.71) = 0.75$$

(c) Similarity matrix with all word alignment scores (best alignment marked in bold) and the alignment similarity for best alignment.



(d) Resulting labeled sentence

Figure 6.3.: Alignments and similarities for the example from Figure 6.2.

4. If the number of lines is equal to the dimension of the matrix, stop. Otherwise go on with the next step.
5. Determine the smallest entry not covered by any line. Subtract this entry from each uncovered row, and then add it to each covered column. Return to step 3.

The zeros cover an optimal assignment over the original cost matrix.

Figure 6.3 contains an example matrix and the resulting optimal alignment for the two example sentences from Figure 6.2.

6.3.7. Label projection

Input to this step is a labeled sentence s , an unlabeled sentence u and the best alignment between them. First, we assign the predicate of the unlabeled sentence as the anchor of the comparison. Then, for every source side argument candidate w_i , we project its label along the alignment onto the aligned candidate on target side $\sigma(w_i)$. If a word w_i is aligned to nothing or has no argument label, nothing is projected.

In the example from Figure 6.3, the word “*larger*” is marked as the predicate and three arguments are added. The label ‘Entity 1’ is projected from the seed sentence argument “*camera*” to the aligned word “*camera*” in the unlabeled sentence. Similarly, ‘Entity 2’ is projected from “*SRLs*” to “*camera*”, and ‘Aspect’ from “*curve*” to “*body*”.

6.4. Analysis of expansion on development set

In this section, we evaluate the structural alignment approach with our adaptations on our task of comparison detection. The first set of experiments is a direct evaluation of the projected labels created by the system on a manually annotated development set.

6.4.1. Data and experimental setup

Seed and test data. As our core labeled seed data set we use our corpus of comparison sentences from English camera reviews (described in Chapter 4). We divide the data into five folds and use one fold as seed data. The rest of the folds is used as test data in the second set of experiments (see Section 6.5).

Expansion data (development set). We want the development set to contain sentences that are possible candidates for the expansion of the seed data, so we cannot just randomly annotate data. Rather, we run a version of the expansion algorithm and take

the expansion sentences proposed by the system to be annotated by a human. We used a relatively small amount of about 50000 camera reviews from `amazon.com` as the expansion data to create the development set (different from the actual expansion set used in the experiments afterwards). We ran the expansion with asymmetric KK candidate creation and used `VS+TREEWINDOW+DEP+POSITION+LEVEL+PATH` as argument similarity. Starting with the first seed sentence, we took the top 3 found expansion sentences per seed until we had 151 different sentences (from 61 seed sentences).

In total we had five annotators who were all graduate students of Computational Linguistics. Each sentence was annotated by three different annotators. The annotators were given a ten-minute introduction to the task. In annotation interface they were given the sentence with the identified predicate marked and had to decide whether the marked word introduces a comparison. If so, they were asked to identify the parts of the comparison. We also asked for a simplified version of the comparison type, annotators had to choose whether the first entity was better (i.e., type `ranked` or `superlative`, direction superior), worse (i.e., type `ranked` or `superlative`, direction inferior), equal to (i.e., type `equative`) or different from (i.e., type `difference`) the second entity.

The whole annotation process took about one hour. Table 6.2 shows some agreement results. For our final development set, we take sentences to be comparative if at least two annotators agree on the decision, otherwise they are non-comparative (even if one annotator disagreed). Out of the 151 predicates marked, 85 were agreed to be introducing a comparison by at least 2 annotators. For arguments, we use lenient text span agreement agr_l , where two spans are considered to match if they have at least one overlapping token. We extract all annotation parts where at least two annotators agree, otherwise no annotation of the type is extracted. In case annotators do not agree on a complete span, only the overlapping part is extracted.

As an example consider the following sentence from the data:

(6.6) *“it’s slightly **bigger** than the *XTi* and the size of the rear screen means buttons have been moved around so it took a little getting used to, but the image quality is very good.”*

All three annotators agree that the marked predicate *“bigger”* introduces a comparison. The annotations for the first entity are *“it”* (twice) and empty (once), so *“it”* is annotated. For the second entity, annotators chose *“the XTi”* and *“XTi”* (twice), so the overlap of all annotators, *“XTi”*, is used. There is disagreement about the aspect, the first annotator left it empty, the second annotator chose *“the size”*, the third *“bigger than”*, so no aspect is annotated. The resulting annotation of the sentence is thus:

(6.7) *“ $[it]_{E1}$ ’s slightly **bigger** than the $[XTi]_{E2}$. . . ”*

Categorical:	3 agree	2 agree	no agreement	A_2	A_3
Comparison yes/no	91	56	4	0.97	0.60
Comparison type	39	39	7	0.92	0.46
Text spans:	3 agree	2 agree	no agreement	agr_{l2}	agr_{l3}
Scale	29	48	8	0.91	0.34
Entity 1	44	40	1	0.99	0.52
Entity 2	59	24	2	0.98	0.69
Aspect	39	40	6	0.93	0.46

Table 6.2.: Inter-annotator agreement on the set of 151 annotated sentences in the development set. Every sentence is annotated by three annotators, we report observed agreement (A) and lenient text span agreement (agr_l) for two ($_2$) and three ($_3$) annotators.

Co-occurrence vectors for vector space similarity. To calculate vector space similarities that are relevant for our domain, we use co-occurrence vectors extracted from a large set of reviews with a total of 40 million tokens. This set includes the above expansion corpus, the electronics part of the HUGE corpus (Jindal and Liu, 2008) and camera reviews from `amazon.com`. Sentence splitting and tokenization is done with Stanford CoreNLP⁴ (Manning et al., 2014). We restrict ourselves to sentence-internal co-occurrence and use a symmetric window of 2 words. Tokens are normalized by lower-casing and non-words are excluded (e.g., numbers, punctuation, URLs, hashtags, emoticons). For the final vectors, we retain the 2000 most frequent dimensions.

Experimental setup and evaluation. To test the various similarity measures and methods for candidate extraction, we simulate the expansion process on the development set. Structural alignment is run for the seed sentences for which we have annotated sentences in the development set (i.e., the first 61 seed sentences) and labels are projected for the top k expansion sentences. If the value of k is such that there are several sentences with the same similarity value and only some of them can be annotated, the selection is done at random, so there will always be exactly k sentences with projected labels.

We compare the projected annotations to the gold standard annotation of the sentences. We report precision, recall and F_1 score on three tasks: predicate identification, argument identification and argument classification. For argument classification scores are micro-averaged over all argument types (P_μ , R_μ , F_μ). For recall, we only compute the recall on predicates that are compatible with the predicate we are currently expand-

⁴<http://nlp.stanford.edu/software/corenlp.shtml>

ing. This serves to exclude predicates that we have no possibility of finding with our approach. So, if a sentence contains an additional predicate that is not compatible with the currently expanded predicate, it and its arguments are ignored. For example if we are looking for expansion sentences for the predicate “*higher*”, the predicate “*superior*” would be ignored in the evaluation of this sentence:

(6.8) “*The D5100 has a far **superior** screen with much **higher** resolution.*”

We compare the different methods of candidate creation (Labeled arguments, FL-SYMM, KK-SYMM, FL, KK, see Section 6.3.3) and different similarity measures for arguments and predicates (Table 6.1, see Sections 6.3.4 and 6.3.5) We compare to a simple **baseline** that just assumes the similarity of 1 for every word pair. This will usually result in the arguments being aligned by the order in which they are put into the candidate list, which is by surface position.

6.4.2. Results and discussion

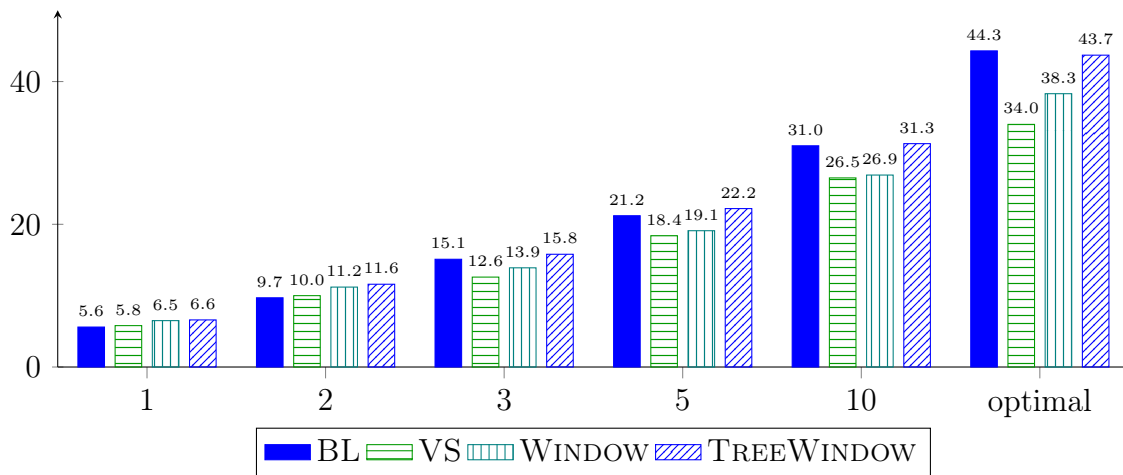
Results using labeled arguments

As a first experiment we use the actual Labeled arguments as argument candidates on both sides. When the predicate candidate on the unlabeled side is not actually a predicate, no argument candidates are extracted on the unlabeled side, which means the unlabeled sentence is ignored. Therefore, all predicates found in the expansion are annotated predicates, so precision on predicates is always 100.

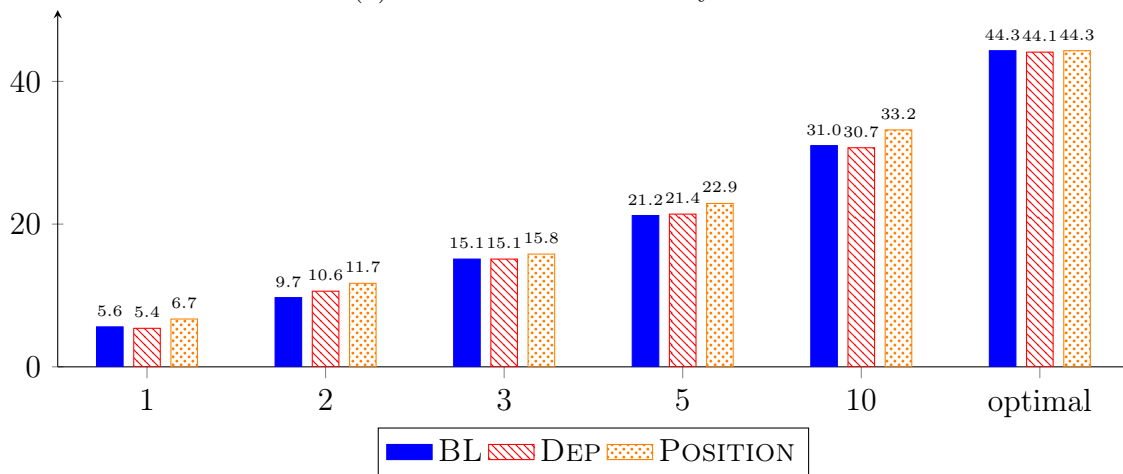
The next decision concerns the number of sentences k to annotate per seed sentence. In our development set, we can set k to the number of expansion sentences there actually are for each seed sentence, i.e., no fixed value of k , but the optimal value for each sentence. When we run this experiment, we get near-perfect recall⁵ and F_1 score. With perfect predicate identification, argument identification on labeled arguments will produce errors only if there is a different number of arguments in source and target sentence. The scores vary slightly for the different similarity measures, but overall we get F_1 scores of about 87. Scores for argument classification depend on argument identification and on the chosen similarity measure and F_1 scores range from 34 to 49.

These scores are a theoretical maximum, as in reality we will not know how many “real” expansion sentences there are. When we fix k to some value that, recall on predicates drops, and this affects all following steps of the pipeline. As an example, for $k = 1$, recall drops to 6.1, with $k = 10$ we are at 46.1. The related drop in recall for argument

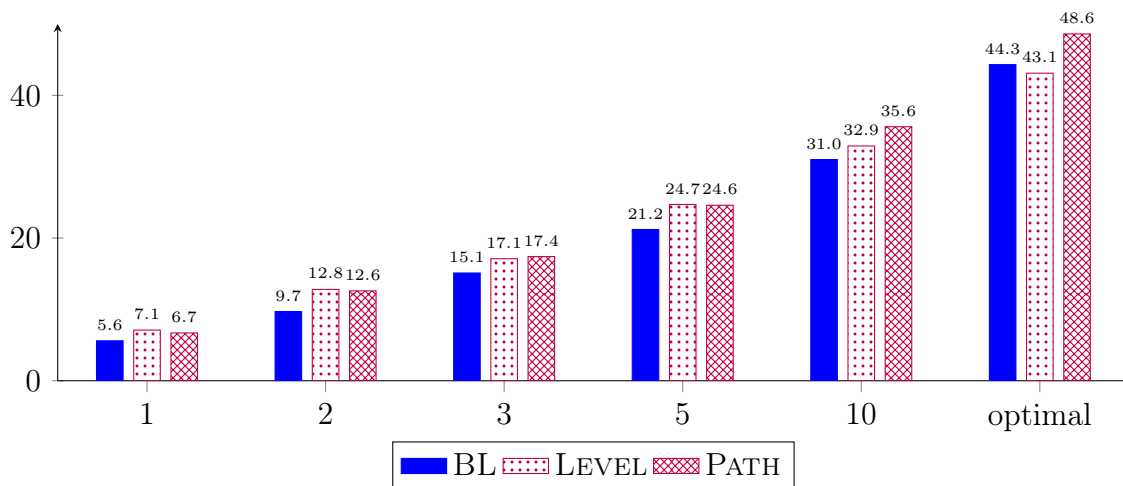
⁵Actual recall is 98.2, because one sentence in the development set has no annotated arguments, so its alignments are discarded because of our rule that at least one role-bearing node must be covered.



(a) Semantic word similarity measures



(b) Flat/surface syntactic word similarity measures



(c) Dependency syntactic word similarity measures

Figure 6.4.: Results (F_1 score for the task of argument classification) on development set for different similarity measures using LABELED arguments on both sides when extracting $k = 1, 2, \dots$ expansion sentences.

identification is in the same range. Again, there are minor differences, but overall the values are the same, independent of the similarity measure that has been used.

In this first set of experiments, we compare results on argument classification for the different argument similarities one at a time. The F_1 scores for different values of k can be found in Figure 6.4 (the complete tables can be found in Table C.1 and Table C.1). We can see that the simple baseline of always assuming a similarity of 1 beats the semantic similarities for $k \geq 3$. Flat and surface syntactic similarities are mostly identical to the baseline. The best results are achieved with PATH similarity.

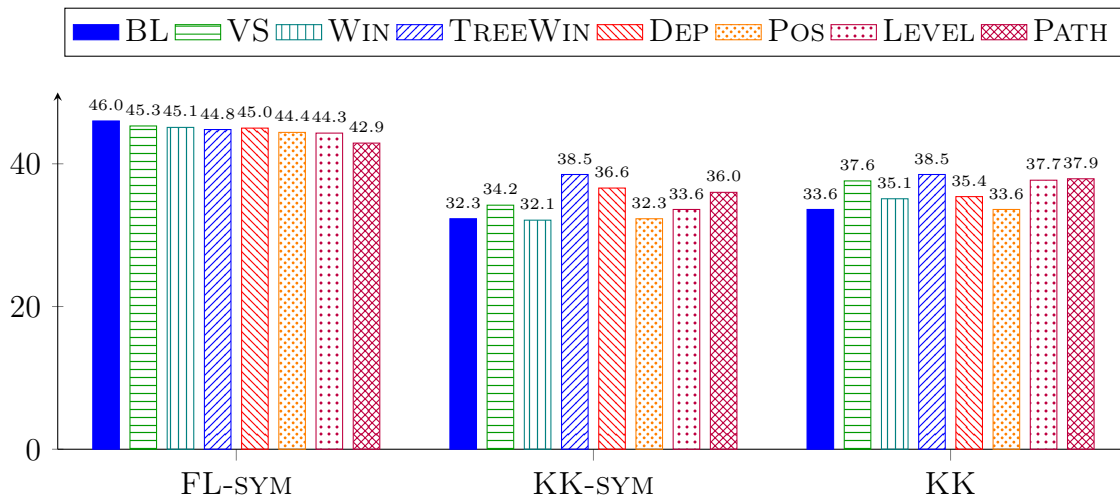
When we run all possible combinations of similarities, few manage to improve over the result by PATH alone. The best performing combination for the optimal number of candidates is TREEWINDOW+DEP+POSITION+LEVEL+PATH with F_1 of 51.36. The combination from the original paper (VS+DEP) is always slightly above DEP alone.

Results using different candidate extraction methods

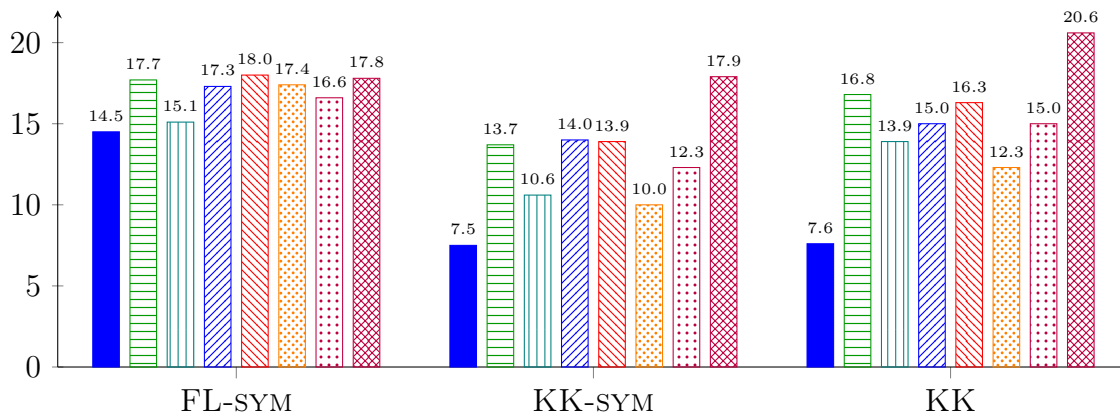
In a realistic setting, the arguments on unlabeled side are unknown, so we need to use our strategies FL or KK for argument candidate creation (see Section 6.3.3). We fix the number of extracted expansion sentences to $k = 10$ and compare both the symmetric candidate creation (i.e., use the method on both sides), and asymmetric candidate creation (i.e., use real arguments on labeled side and the method only on unlabeled side). Symmetric extraction is marked with the suffix -SYM.

The results of the experiments can be found in Figure 6.5 (the complete tables are in Table C.3, FL is omitted for better readability, as it is very similar to FL-SYM). Not surprisingly we observe a drop in overall numbers compared to using only the actual arguments. Also, differences between the settings are more pronounced due to the possibility of choosing candidates that are not actually predicates or arguments.

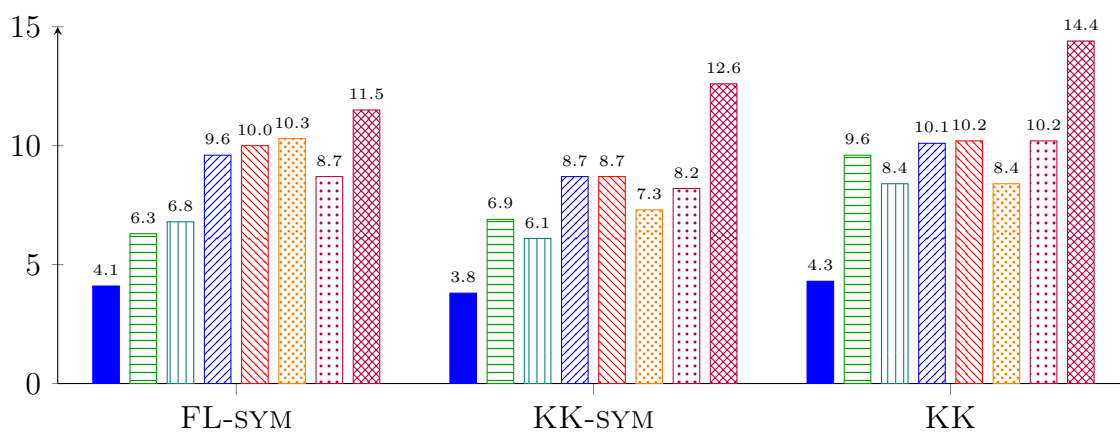
The FL-SYM method searches for the exact same path between predicate and argument in both sentences. As we find many such paths only once in our labeled data, we expected the method to suffer from low recall. This is not the case, recall for FL-SYM is the same or even a bit higher as for KK-SYM candidate creation. This is mainly due to the fact that the direct dependents of the predicate are always argument candidates. Out of 4500 checked paths, only 376 are found, but still the candidate set is empty for only 1016 sentences. In total, 850 expansion sentences are extracted and the average size of the non-empty candidate sets is 1.8. For comparison, with KK-SYM candidate creation, the candidate set is never empty, the average size is 9.7 items and 955 expansion sentences are extracted.



(a) Results for the task of predicate identification



(b) Results for the task of argument identification



(c) Results for the task of argument classification

Figure 6.5.: Results (F_1 score) on development set for different similarity measures and different candidate extraction methods (symmetric FL-SYMM, KK-SYMM or asymmetric FL, KK) when extracting $k = 10$ expansion sentences.

Performance differences between the individual similarity measures are largest in argument classification. Despite errors being propagated through the pipeline, it is not necessarily the highest performing similarity in predicate identification that has the best argument classification scores. On the contrary, for FL-SYM candidate creation, on predicates the BASELINE outperforms every other similarity measure, but is in turn always outperformed on argument classification. On predicates, FL-SYM easily outperforms KK-SYM, but the scores on argument classification are similar. On arguments, the PATH similarity outperforms every other similarity measure, similar to the results on the labeled arguments. The difference is more pronounced for KK-SYM candidate creation, where we have larger candidate sets and context is more important to pick out the correct candidate. When using asymmetric candidate creation, all numbers improve (for better readability FL is not shown, but is very similar to FL-SYM, numbers are included in Table C.3). The patterns of differences between the similarities stay the same. The overall improvement is mainly due to increased recall.

When we run all 127 possible combinations of similarities, there is no one best combination for all tasks, but different combinations are very close together. For FL-SYM the combination from the original SRL work (VS+DEP) scores best on predicate and argument identification. For argument classification it is somewhere in the middle, but with only 1.2 points difference in F_1 score to the top result. For asymmetric candidate creation, VS+DEP is still among the top scoring combinations in predicate and argument identification and in the middle for argument classification. The top scoring combinations for FL are VS+WINDOW+DEP for predicate identification (F_1 score of 46.01), VS+TREEWINDOW+DEP+LEVEL and VS+DEP+LEVEL for argument identification (F_1 score of 21.95), and DEP+POSITION (F_1 score of 14.34) for argument classification. Many other combinations of similarities are very close to these scores.

The situation is similar for KK-SYM and KK candidate creation. Asymmetric candidate creation is better than symmetric creation and different combinations of similarities perform best on all tasks for both cases with little performance differences. The top scoring combinations for KK are VS+TREEWINDOW+DEP+LEVEL for predicate identification (F_1 score of 41.21), VS+WINDOW+TREEWINDOW+DEP+LEVEL+PATH and VS+WINDOW+TREEWINDOW+DEP+POSITION+LEVEL+PATH for argument identification (F_1 score of 22.38), and VS+TREEWINDOW+DEP+POSITION+PATH (F_1 score of 16.7) for argument classification. VS+DEP is always somewhere in the middle.

In summary, the asymmetric method of candidate creation is better than the symmetric method. There is no one combination of similarity measures that clearly outperforms all others for argument candidate selection. The ‘flat’ list of similarities VS+DEP, is

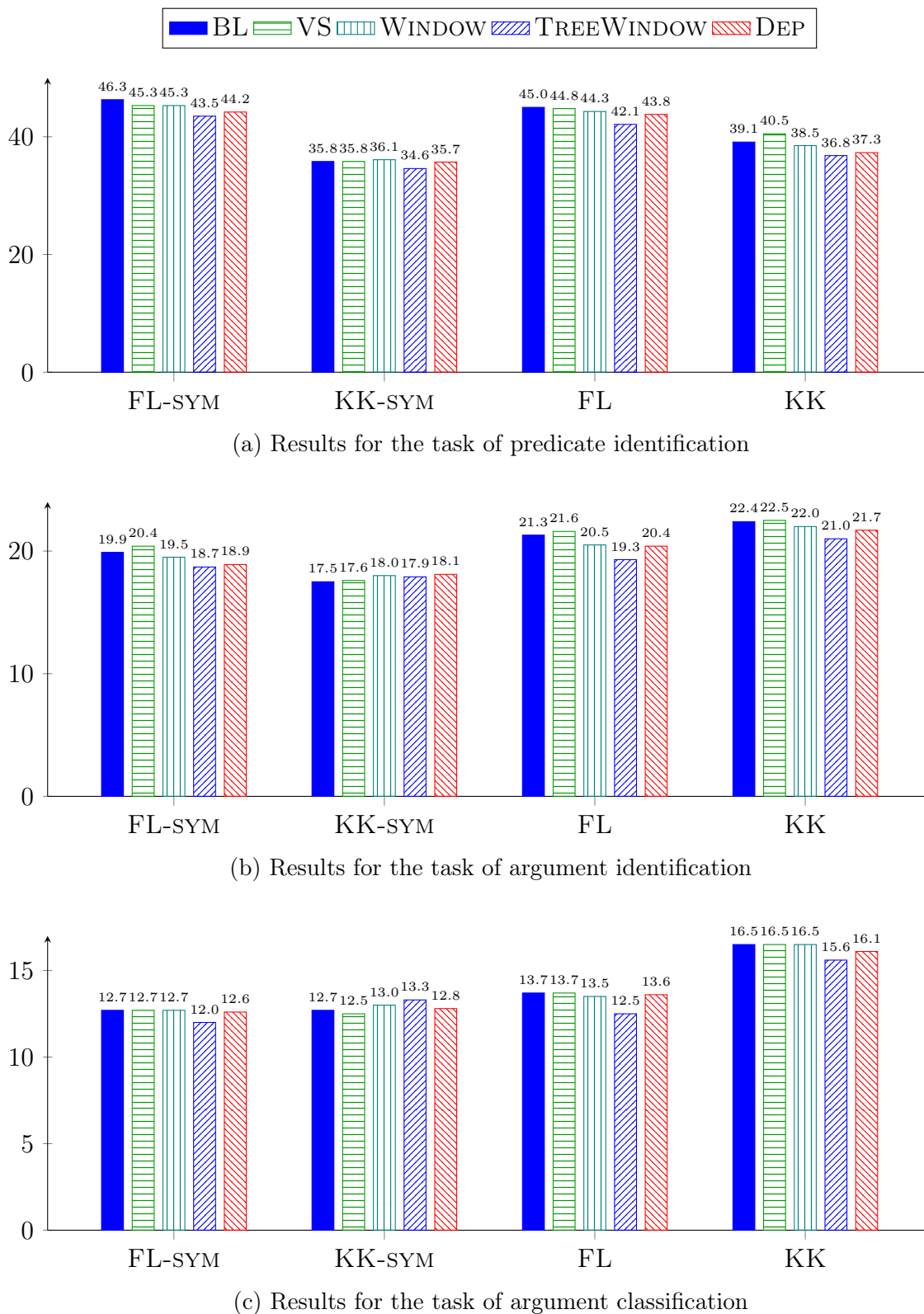


Figure 6.6.: Results (F_1 score) on development set for different predicate similarity measures and candidate extraction methods (symmetric FL-SYMM, KK-SYMM or asymmetric FL, KK) when extracting $k = 10$ expansion sentences.

not a bad choice when the candidate sets are small. When the sets get larger, as is the case with our KK candidate extraction, some combination of more similarity measures is needed. It is generally necessary for a good combination to include at least one semantic and one syntactic similarity measure, but the exact combination of measures may not make a relevant difference.

Results using predicate similarities

Our last experiment on the development set concerns the influence of predicate similarities. Like in the previous experiment, we fix the number of extracted expansion sentences to $k = 10$ and compare both the symmetric and asymmetric candidate creation. For this experiment we use the argument similarity measure that combines all individual measures: VS+WINDOW+TREEWINDOW+DEP+POSITION+LEVEL+PATH.

The results of the experiments can be found in Figure 6.6 (complete results in Table C.4). There are no major differences between the candidate creation approaches or symmetric and asymmetric methods. In general, the effect any predicate similarity measure has is at best marginally positive and sometimes slightly harmful. Among all possible combinations none beats the baseline by more than a very small margin.

6.5. Experiments on training set expansion

The second set of experiments is an indirect evaluation. We add sentences extracted with structural alignment to the training data of our system CSRL and evaluate the effect on test set performance.

6.5.1. Data and experimental setup

Seed, test and expansion data. As our core labeled data set we use our corpus of comparison sentences from English camera reviews (described in Chapter 4). We divide the data into five folds and use one fold as seed data and the rest as test data. The full **seed data** contains 342 sentences with 415 predicates. The **test data** contains 1365 sentences with 1693 predicates.

As the unlabeled **expansion data**, we use a set of 280000 camera review sentences from `epinions.com`. This is the remainder of the data from (Branavan et al., 2009)⁶ that has not been annotated in our corpus. The reviews are split into sentences and

⁶<http://groups.csail.mit.edu/rbg/code/precis/>

tokenized with Stanford CoreNLP⁷ (Manning et al., 2014), and parsed with the MATE parser (Bohnet, 2010). Note that expansion sentences are never used in testing, we always only test on human-annotated data. We use the same **co-occurrence vectors** for vector space similarity as in the previous set of experiments.

Experimental setup. This set of experiments is an indirect evaluation that does not look at the projected labels themselves, but only at the influence the added training data has on the classification performance of our comparison detection system. We use our system CSRL described in Chapter 5 to detect comparisons. We use the same settings and same features for all experiments, namely the SYNTAX setting, which performed best in our experiments in that chapter.

As a **baseline**, we train CSRL on the seed data only. To evaluate whether the found expansion sentences are useful, we add the k best expansion sentences per seed predicate to the training data and train on this expanded corpus. We use the test data for evaluation and compare classification performance of training on the expanded seed data to the baseline. We report precision (P), recall (R), and F_1 score (F) on three tasks: predicate identification, argument identification and argument classification. Argument classification scores are micro-averaged over all argument types (P_μ, R_μ, F_μ).

Compared versions of structural alignment. We have three main decision points in the setup of structural alignment: argument candidate creation (see Section 6.3.3), argument similarities (see Section 6.3.4), and predicate similarities (see Section 6.3.5). For argument candidate creation, we compare the two methods FL and KK. We restrict ourselves to the asymmetric extraction, i.e., we use real arguments on labeled side and the method only on unlabeled side. In the development set experiments, this always outperformed symmetric creation.

As there is a huge number of possible combinations of argument similarity measures which we cannot possibly investigate, we restrict ourselves to two combinations. The first combination combines FLAT similarities which do not consider any context (VS+DEP). This corresponds to the similarity used by Fürstenau and Lapata (2009, 2012). CONTEXT similarities is the best combination of similarities from our development set experiments which includes surface and dependency context information, i.e., the combination of all similarities (VS+TREEWINDOW+DEP+POSITION+LEVEL+PATH). For predicate similarities, we first follow the original setup and use no predicate similarity. In the last experiments, we use VS+WINDOW+DEP as predicate similarity (PREDS setting).

⁷<http://nlp.stanford.edu/software/corenlp.shtml>

To summarize, we test the following versions of the expansion:

- FL-FLAT: asymmetric *FL* argument candidate creation, *flat* argument similarity, no predicate similarity (the setting closest to the original setup for SRL).
- KK-FLAT: asymmetric *KK* argument candidate creation, *flat* argument similarity, no predicate similarity.
- FL-CONTEXT: asymmetric *FL* argument candidate creation, *context* similarities, no predicate similarity.
- KK-CONTEXT: asymmetric *KK* argument candidate creation, *context* similarities, no predicate similarity.
- FL-CONTEXT-PREDS: asymmetric *FL* argument candidate creation, *context* argument similarities and *predicate* similarities.
- KK-CONTEXT-PREDS: asymmetric *KK* argument candidate creation, *context* argument similarities and *predicate* similarities.

Research questions. There are two main questions we investigate:

1. How many seed sentences should be used (varying d)?
2. How many expansion sentences should be used per seed (varying k)?

We expect that the training data expansion is helpful in low-resource and high-precision settings (i.e., d and k are small). This corresponds to a scenario where only a limited amount of sentences has been annotated for a new application domain or a new language. We consider this to be a more realistic scenario for our task than the one used in Fürstenu and Lapata (2012), where a fixed number of training examples per frame is used. In contrast to SRL, we do not expect to know predicates or frames for comparisons in advance.

6.5.2. Results

It is impossible to present all results from all experiments as we have three tasks, six systems plus baseline and many values for k and d . We limit the following discussion to argument identification, but the results for the other tasks are similar.

Figure 6.7 shows some results for argument identification on the test set in terms of F_1 score. The different curves represent expanding and training on different percentages d of the seed set, from 10% (only 34 seed sentences) to 100% (full set). The x-axis shows k , the number of expansion sentences added per seed sentence. The value 0 corresponds to the baseline, i.e., training on the seed sentences only. Results for a selected number of settings for d and k can be found in Tables C.5 to C.10.

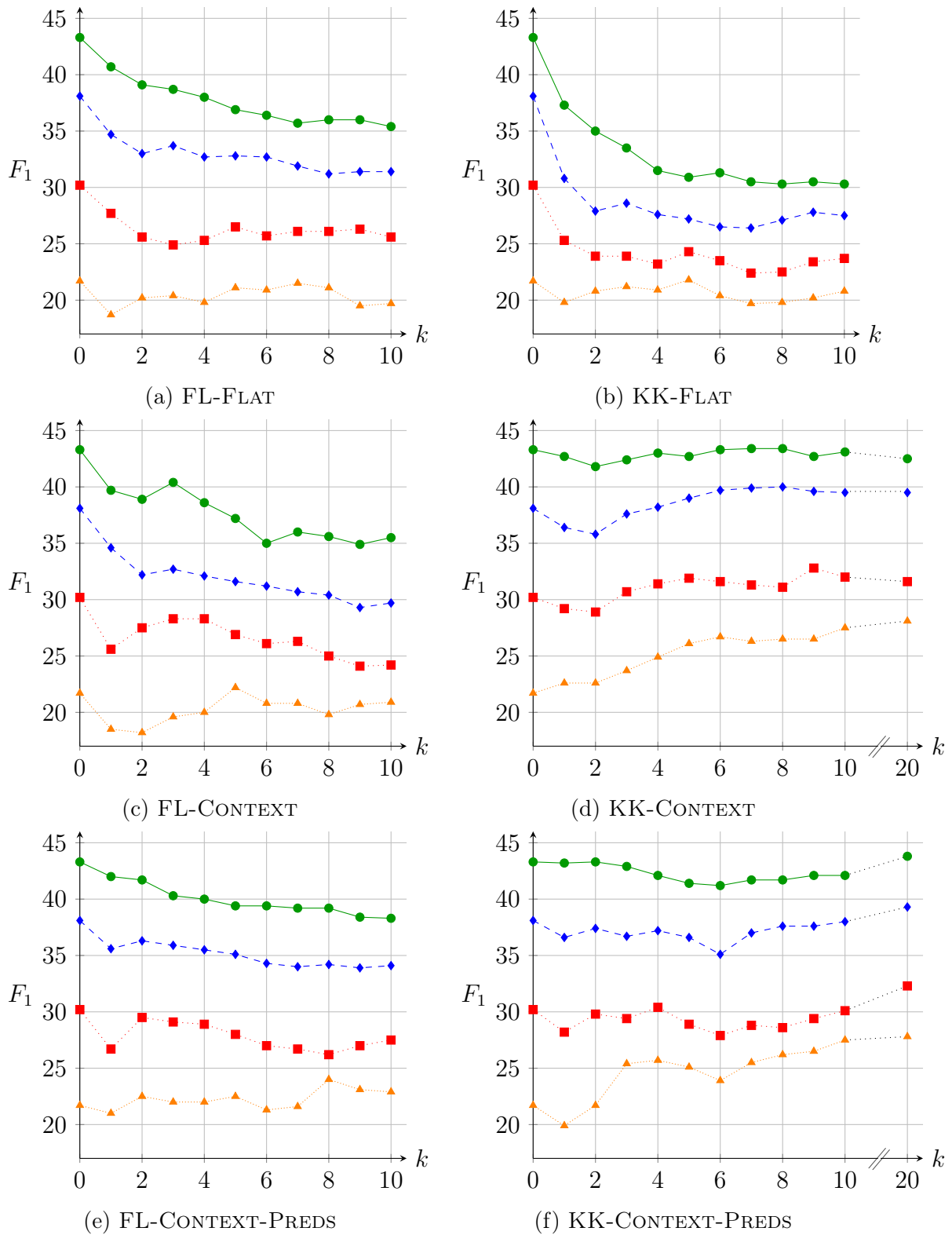


Figure 6.7.: F_1 score for argument identification on test set varying the number of expansion sentences k and the different percentages d of the seed data (curves from top to bottom: 100%, 50%, 25%, 10%).

We see that FL-FLAT, the setting from the original SRL approach, does not manage to find useful expansion sentences. No setting of k manages to improve over the non-expanded baseline, every added expansion sentence only decreases performance. When we use our dependency-based candidate selection method (KK-FLAT), results get worse. This is expected as our method creates many more argument candidates, so more incorrect assignments can be made when not enough context information is available to distinguish between them. In line with the results reported for SRL, for most cases as k gets larger, performance drops, because the amount of introduced noise outweighs the benefits of additional training data.

When we use contextual argument similarities, results improve over using flat similarities. With FL-CONTEXT, the difference in performance over using flat similarities is not very big. Adding expansion sentences still give results below the non-expanded baseline. A possible explanation is that the created argument candidates are already very precise and the paths-based creation method includes the similar information to what contextual similarity provides. With KK-CONTEXT, the improvement compared to using flat similarities is considerable. Especially for low values of d , we manage to get a small improvement over the non-expanded baseline. After $k = 10$ (shown with dotted lines) the curves flatten.

To illustrate the sentences selected by the different systems, consider this example:

- (6.9) a. “I felt **more** [comfortable]_S with [XTi]_{E1}” (seed)
 b. “I bought this because my wife didn’t feel [comfortable]_S with all the features/functions of the **more** complex [C5050Z]_{E1}.” (KK-FLAT)
 c. “I was much **more** [comfortable]_S with the [DSC-S75]_{E1}” (KK-CONTEXT)

Sentence 6.9a is the seed sentence, Sentence 6.9b is the sentence selected as first expansion sentence by KK-FLAT, Sentence 6.9c is the one proposed by KK-CONTEXT. While aligning “comfortable” in Sentence 6.9b with the labeled aspect seems like a perfect match in isolation, Sentence 6.9c is a much better choice in context.

When we add predicate similarities, FL-CONTEXT-PRED manages to improve a bit over FL-CONTEXT, but results are still below the non-expanded baseline. The performance of KK-CONTEXT-PRED for k between 1 and 10 is similar to KK-CONTEXT. Looking at the expansion sentences, both systems often select the same sentences with the same alignments as expansion sentences. Interestingly, for values of $k > 10$, performance improves over $k = 10$ to values above the non-expanded baseline. Figure 6.8 contains more detailed curves for KK-CONTEXT-PRED with k between 0 and 30. Scores continue to rise, although the curves get flatter when k increases. Looking at the newly extracted expansion sentences, there is no big decrease in the quality of these sentences

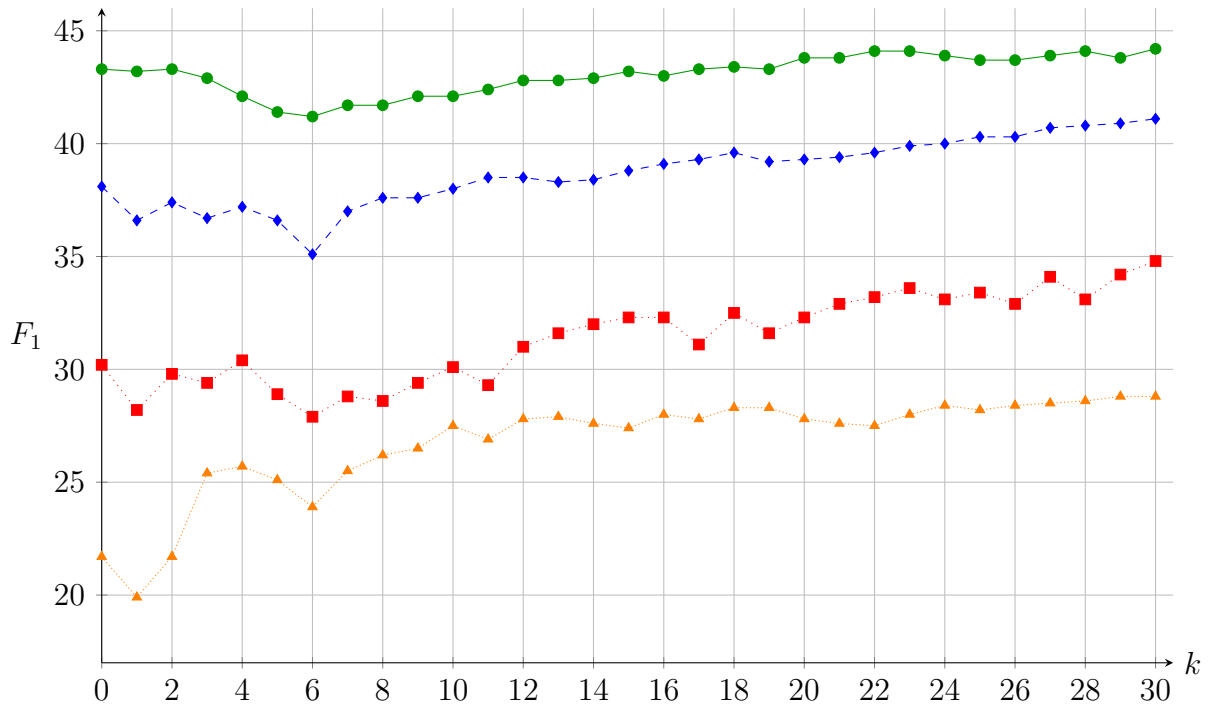


Figure 6.8.: F_1 score for argument identification on test set for KK-CONTEXT-PREDS varying the number of expansion sentences k and the different percentages d of the seed data (curves from top to bottom: 100%, 50%, 25%, 10%).

compared to the first 10 expansion sentences. Often, the alignment scores are close together. It is understandable that performance does not suffer when more equally good sentences are added to the training data. Why performance increases and not only stabilizes remains unclear.

Figure 6.9 shows learning curves for argument identification for each system with the best setting for k (usually 1, 10 for KK-CONTEXT, 10 and 30 for KK-CONTEXT-PREDS). All systems except KK-CONTEXT and KK-CONTEXT-PRED are below the baseline. The best value of k for KK-CONTEXT in our experiments is 10, which is shown in the graph. The results are very similar for all $k \geq 5$, for lower values of k , the results drop below the baseline. The best setting manages to improve over the non-expanded baseline in low resource settings (small d), but the curves get closer to each other when more seed data is added and the effect disappears at the end. The best value of k for KK-CONTEXT-PRED in our experiments is 30, for comparison to KK-CONTEXT, we also show the results for $k = 10$. For $k = 10$, KK-CONTEXT-PRED performs mostly worse than KK-CONTEXT and comparable to the baseline. For $k = 30$, there is quite a large improvement over the baseline for small values of d , but the gap gets smaller as d gets larger and the effect nearly disappears at the end.

As a summary, with the FL candidate creation, no setting produces useful expansion sentences that improve the performance of a system trained on the expanded data over a the baseline of a system trained on non-expanded data. With our KK candidate creation and contextual similarities, we manage to improve over the baseline. The best result of our experiments is obtained by the system KK-CONTEXT-PREDS using $k = 30$ neighbors. The improvement is particularly noticeable for small values of d . Comparing our results to our expectations, we were able to validate that training data expansion is helpful in low-resource settings. However, the context-aware systems do not perform as well in a high-precision setting (small k) as expected, but it seems the addition of many expansion sentences is beneficial.

6.5.3. Discussion

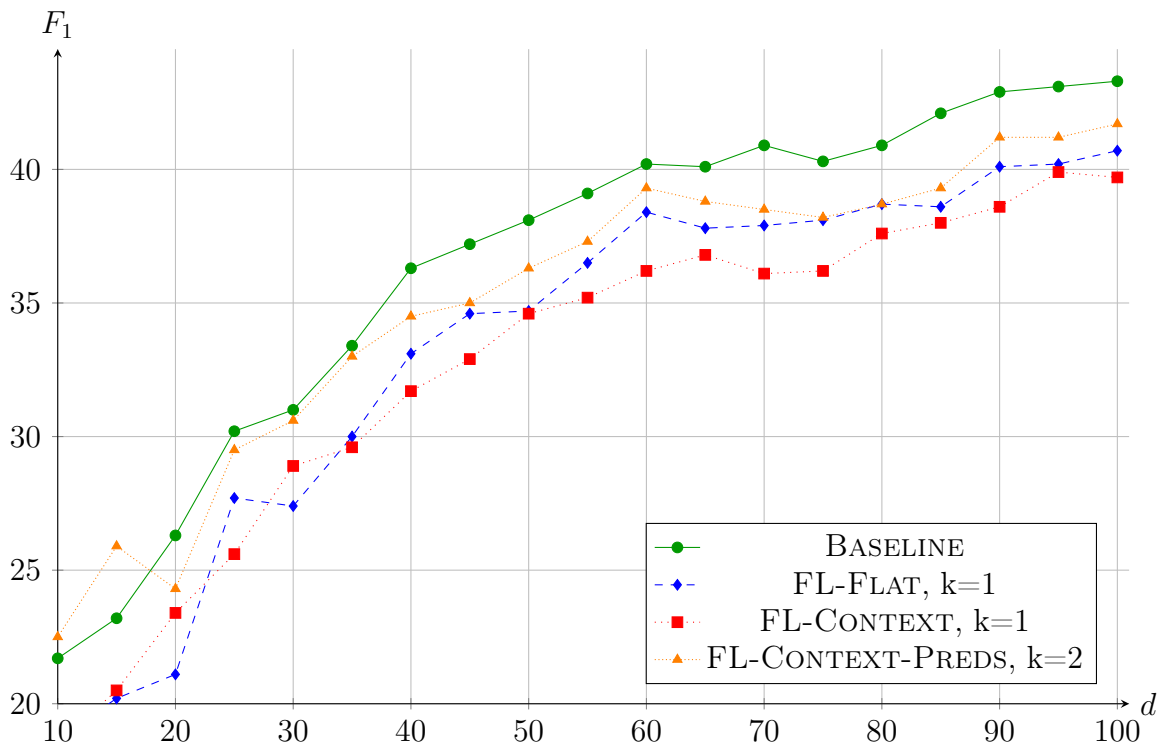
While manually inspecting the extracted sentences, we see very good matches, but also many label assignments that are very wrong. We can identify two main issues with the extracted sentences, that we discuss in this section.

The first issue that affects all sentiment-related tasks is subjectivity. Often sentiment words (or in our case comparison words) appear in non-sentiment (non-comparative) contexts, but these contexts are very hard to distinguish from sentiment contexts. Consider these example sentences:

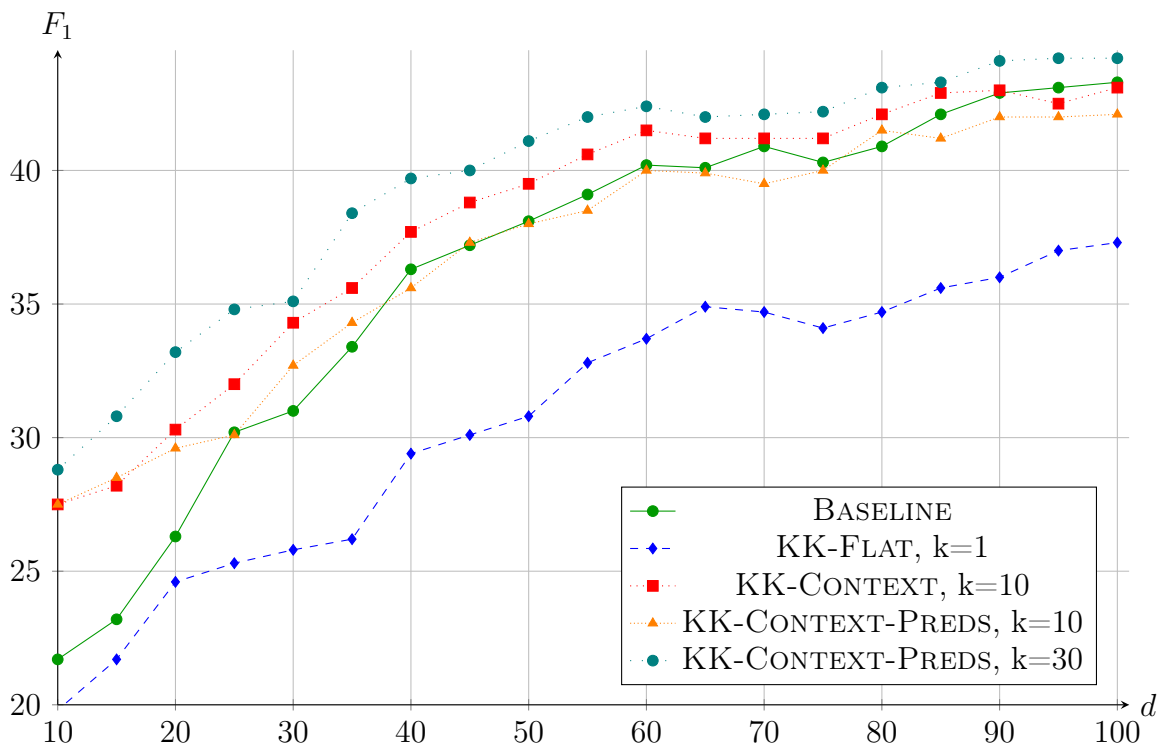
- (6.10) a. “*This is largely a function of the much **smaller** [SD media]_{E1}.*” (seed)
 b. “*... I can take anywhere from 90 (High Resolution Images) to hundreds of differing **lower** resolution [images]_{E1}.*” (KK-CONTEXT)

Sentence 6.10a is the seed sentence, Sentence 6.10b is the best sentence selected by the context-aware system KK-CONTEXT. Though the two phrases “*smaller SD media*” and “*lower resolution images*” are a good match, the word “*lower*” in sentence Sentence 6.10b does not introduce a product comparison. Instead, it describes a type of picture that you can take with the camera. Such uses are relatively frequent and often mistakenly chosen as expansion sentences. Such “false positives” mainly affect predicate identification, but errors in this first step are propagated through the pipeline.

We have attempted to improve predicate identification by adding predicate similarity to our approach. Indeed Sentence 6.10b is not among the top expansion sentences anymore when we use KK-CONTEXT-PRED. There are cases, like this example, where such a selection works, but many of the distinctions cannot be picked up this way as they depend on world knowledge or very subtle cues. One way to further address this



(a) Best systems with FL argument candidate creation.



(b) Best systems with KK argument candidate creation.

Figure 6.9.: Learning curves (F_1 score for argument identification) with different percentages d of the seed data for the best value of k for each system.

challenge would be to include a separate subjectivity analysis step into the processing to filter out such non-comparative usages.

A second type of error is caused by the non-aligned part of the chosen expansion sentences. Expansion sentences may contain other predicates besides the expanded predicate. Consider this (short) example:

- (6.11) a. “*That said, the **larger** LCD [screen]_A is really an improvement.*” (seed)
b. “*The **smaller** 2-inch [screen]_A has higher resolution of 118,000 pixels!*”
(KK-CONTEXT)

Sentence 6.11b is selected as an expansion sentence for seed sentence Sentence 6.11a by KK-CONTEXT, and the first part it is not a bad fit. Unfortunately, the additional predicate “*higher*” in the expansion sentence can not be detected, thereby creating a “false negative” example for predicate identification. This happens quite frequently, as the sentences in our expansion data are often very long. This issue could be addressed in several ways, e.g., by cutting out only the relevant part of the sentence that contains the projected arguments, by simplifying expansion sentences or by a filtering step where only “simple” sentences are kept as possible expansion sentences.

6.6. Summary

In this chapter we have investigated whether structural alignment can be adapted to the task of detecting comparisons. Structural alignment is a semi-supervised method that has been successfully used for projecting SRL annotations from a small seed set of labeled sentences onto a large set of unlabeled sentences. Capturing the structure of comparisons is the key step in this process to ensure that the found sentences are similar enough so that the projection of labels onto them produces correctly labeled new training examples, but also different enough to include linguistic variations. We have presented some adaptations of the method to our task of comparison detection thereby tailoring the approach to the structural characteristic of comparisons. We have evaluated the projected labels directly on a development set. Additionally, we evaluated the success of the method by comparing the performance of a system trained on the original data with the performance of a system trained on the expanded data. We have found that our adapted structural alignment can successfully be applied and improves over a non-expanded baseline in low-resource settings, e.g., when only a very small amount of training data in the desired domain or language is available.

7. Towards the prediction of product rankings based on comparisons

Most of the work presented in this chapter has been published in (Kessler et al., 2015). This is joint work with Roman Klinger who contributed his system JFSA, the review data used for the evaluation, and the idea of evaluating against the Amazon sales ranking.

7.1. Introduction

In the preceding chapters we have discussed comparisons and methods to address them computationally. Our system CSRL, presented in Chapter 5, detects comparisons and their relevant components, i.e., what products are compared, in what aspect and which of the products is rated as better. The application of this system to a large amount of product reviews will result in a large number of individual comparisons. While this information by itself may be interesting for someone who is doing a thorough analysis of all expressed opinions, e.g., a social media analyst, it is not suitable for typical situations where users turn to sentiment information to support a process of choice. In this scenario, out of a range of potentially suitable items, the user needs to make an informed decision about which product to select. One step towards reaching a decision is to rank the products in question according the product aspects that are relevant for the user. Textual comparisons provide explicit comparative assessments between products, which makes them presumably the most useful kind of expression for this task.

In this chapter, we work on the task of producing a ranked list of products that complements the isolated prediction of ratings. While the isolated prediction of ratings for a product or different aspects of a product helps the user to find a selection of suitable products, the number of different aspects to consider for a complex product will usually make it difficult to make a final choice for the product that best meets the

specific needs of the user. Also, while measurable aspect such as price or weight may be compared based on the technical data provided by the manufacturers, reviews are more suitable to compare “soft” aspects such as usability or the behavior of a product in special circumstances. In this situation, a ranked list of products that is based on the experiences and evaluations of previous users as they are expressed in reviews may be a valuable resource to support the user in this process of choice, especially if it is transparent what textual data has been used to produce the rating. Unfortunately a popular product will have a large amount of reviews so that reading them all may be impractical for a human. Using comparisons as a basis for the ranking has the advantage that the reviewer’s basis for coming up with a comparative assessment is explicitly stated and can directly be shown to the user.

To properly define the task of producing a ranked list of products, we need to address two fundamental issues: The selection of a gold ranking used for evaluation and methods that can be used to create such rankings. For the first issue, the selection of a gold ranking, we describe three possible sources for external gold standard rankings: a sales ranking, which reflects the number of times an item has been sold in a given time period relative to similar products, an expert ranking, for which a domain expert compares different products and ranks them by their quality, and a crowdsourcing-based annotation of rankings, where quality rankings are made by different users. These rankings differ in the criteria used to create the ranking, their availability and the usefulness for an end user. For our experiments we use two external gold rankings for cameras, the sales ranking from amazon.com and the expert ranking from the website snapsort.com, a service that collects detailed information about cameras and provides comparisons between them. As these two rankings are conceptually very different, it is interesting to see how these differences are reflected in the experimental results.

The second issue concerns the methods used to automatically create product rankings based on review data. It is a straight-forward idea to exploit textual comparison expressions which occur frequently in reviews and are sometimes easier to provide than absolute judgments about an entity. Also, comparative rankings can be interpreted more consistently, e.g., “*A is better than B*” is explicit about the ordering of the entities, whereas it depends on many cultural and personal factors whether “*amazing*” or “*not bad*” is chosen to describe a good product. We use our comparison detection system CSRL to detect comparisons. The individual comparisons from different reviews are then aggregated to create a ranking over all products. Specifically, the ranking system we present in this chapter calculates a score for each product based on the number of times a product is mentioned as the preferred or non-preferred entity in a comparison.

We evaluate the performance of comparison-based ranking compared to other ranking methods based on subjective phrases or review meta data.

As an additional point, we investigate aspect-specific rankings. While overall rankings of products may have some uses, most of the time the user will be interested in rankings that place particular importance on a few specific aspects of a product and ignore other aspects. For example, if we have statements that “*A has a better lens than B and C*”, “*B has higher resolution than C*” and “*B is heavier than C*”, we can rank the products according to the user’s preferences as A, B, C if the resolution is more important than the weight or A, C, B in the reverse case. CSRL extracts information about which aspect is being compared together with the compared entities. We can use this information to create aspect-specific rankings which consider only those expressions that refer to a specific aspect of the product to produce the rankings. A ranking from a combination of selected aspects can be used to create specific, personalized rankings and aspect-specific rankings can also be used to determine the influence of an aspect on the overall ranking. We present some preliminary experiments to investigate the impact of different aspects on the produced rankings.

To summarize, this chapter addresses the last of our research questions, research question C, which has been the motivation for our work on comparisons:

Research Question C: *How can information found in textual comparisons be used to support a process of choice?*

More specifically, this chapter explores the following questions:

- What are possible sources for an external gold standard ranking of products?
- How can methods for comparison-based opinion mining be used to predict product rankings?
- How do comparison-based methods perform on external rankings compared to methods that use subjective phrases or review meta data?
- Can we produce aspect-specific rankings that allow for an understanding of the impact of each aspect on the global ranking?

7.2. Related work

Previous work on ranking products by exploiting sentiment information is comparatively limited, although the task is of interest also in the areas of e-commerce and information retrieval. Especially in e-commerce the importance of reviews and other user-generated content on sales has been widely discussed (Zhang et al., 2013).

One example for an information retrieval approach is Ganesan and Zhai (2012), who cast the task of entity ranking as the problem of retrieving the best-fitting opinion document for a query about a product. Each opinion document is a concatenation of all reviews about a product. They enhance their IR models by splitting the query into separate parts for the product's aspects and use dictionary-based query expansion of opinion words. They evaluate on hotel and car reviews from `epinions.com` within an information retrieval framework, where they collect typical user queries, use the star ratings from reviews to generate relevance judgments, and evaluate whether the relevant entities for each query are found by their system. While such approaches are interesting, they are conceptually very different from sentiment analysis.

Kurashima et al. (2008) work on the task of ranking products in a way that is closer to the sentiment analysis paradigm. For a Japanese query about one product, they retrieve competitor products and a ranking between them. Their approach is based on textual comparisons and link analysis motivated by the idea of modeling a potential customer that moves through a graph of products searching for the best product. The rank of a product can be interpreted as the likelihood that a potential customer will buy this product. The ranking of products is based on a product graph constructed from the identified comparisons. Each entity is a node in the product graph and there is a directed edge between two nodes if a comparative relation is found between the entities. The direction of the edge is from the non-preferred product to the preferred product in the comparison, if both types of comparisons are found, two edges are introduced. Each edge is assigned a weight that represents the percentage of comparisons between source and target entity in which the target entity is evaluated as better. The ranking of entities is done by computing "graph centrality" with a link analysis algorithm similar to PageRank (Page et al., 1999) with special consideration for the edge weights. They perform some experiments on blog posts about movies and visualize the ranking results, but do not evaluate ranking performance.

Zhang et al. (2009, 2010) present a similar approach that uses both subjective sentences and comparative sentences. Like Kurashima et al. (2008), they construct a product graph with products as nodes and relations as edges based on the results of comparison detection. Product nodes are assigned a weight according to the ratio of positive to negative sentences. The weight of an edge is the ratio of positive versus negative comparisons between the two products. For ranking they use some variant of PageRank that takes into account node and edge weights. Zhang et al. (2009) work on data from `amazon.com` in the domain of digital cameras and TVs, and evaluate their rankings against expert rankings from `smartratings.com`. As measure of success they report the

overlap between their 10% top-ranked products and the products in the expert ranking, distinguishing buckets for different price ranges. They find an overlap of about 60%. The motivation for this measure is unclear, they reason that only very few products are reviewed by the experts, but it is not clear why the products reviewed by experts should correspond to the top ranked products. They also compare their rankings to the sales rank from `amazon.com`, but find no correlation. In their follow-up work, Zhang et al. (2010) introduce aspect-specific rankings and compare them to the general ranking by measuring the overlap between the 10% of highest ranked products when using all data and the top 10% when considering only one aspect. Overlaps are high, meaning that aspect-specific rankings do not differ much from the general rankings. Still, different aspects are of varying importance for the overall rating. For the digital camera domain they report that the aspects “lens” and “resolution” are those with the highest influence¹. They do not compare the aspect-specific rankings to an external ranking.

Li et al. (2011) also present an approach that uses comparisons and subjective sentences with link analysis. They build a product graph, edges are directed towards the entity that more users prefer in comparisons, the weight of an edge incorporates polarity information and comparison information. Graphs are created separately for each of four different high-level aspects (design, feature, performance, ease of use). The mapping of individual subjective sentences and comparisons to these high-level aspects seems to use *k*-means clustering. They do not create an overall ranking for all aspects. Like Kurashima et al. (2008), they use an algorithm similar to PageRank to assign a score to each product. They work on reviews about phones and mp3 players and evaluate their system against the results of product comparisons offered by two review websites. They present all rankings to domain experts and ask them to judge which one they prefer. In most cases, the ranking produced by their system is preferred. They report that adding information from community-based question-answer pairs further improves results.

Zhang et al. (2013) also propose an approach that uses comparisons and product graphs. They experiment with different ways of creating edges between the nodes: One link that represents the direction of the preference of most users, one link each for positive and negative, one for each found comparison. To calculate ranking scores, they compare PageRank and a similar link-based algorithm. The data they work on is from camera reviews and they evaluate against the sales rank from `amazon.com`. Spearman’s rank coefficients for their systems are in the range of 0.1 – 0.33, the worst result being the baseline that uses average star rating, the best result coming from the baseline that

¹the aspect *resolution* is mapped to the words “*resolution*”, “*pixel*”, “*megapixel*”, the aspect *lens* to “*lens*”, “*wide angle*”, “*normal range*”.

uses the number of reviews. Their methods are somewhere in the middle, the differences between the different methods for link creation are not very big. In terms of the used gold ranking and evaluation methodology, this is the approach most related to our work.

Most recently, Tkachenko and Lauw (2014) propose an integrated approach that jointly works on two levels, the sentence-level where an individual comparison is found, and the entity-level which aggregates the sentence-level comparisons to relations between entities. The intuition is that both levels help each other: knowing that an entity is evaluated as better as than another one due to transitivity helps the detection of individual comparisons. They create separate rankings for four high-level aspects (functionality, form factor, image quality, price). They work on camera reviews and perform two types of evaluations. As an internal evaluation, they compare their system output to the ranking that results from aggregation over gold standard comparisons which have been annotated by crowdsourcing. As external evaluation, they generate measurements of product quality for specific predefined characteristics that can be extracted from structured data (e.g., that smaller is better for cameras). Among other baselines, they compare to the one proposed by Kurashima et al. (2008) and report a substantial improvement over the baselines.

In contrast to the above approaches, we do not use graph-based methods for ranking, but directly aggregate subjective phrases and comparisons to produce ranking scores. While most of the previous systems use comparisons for their approach, they address the detection of these comparisons in a rather ad hoc way with word lists and heuristics. Our system for comparison detection is much more sophisticated and our method is not limited to sentences that contain two entities or to comparisons that involve a specific aspect. We evaluate against two external gold standard rankings and use established evaluation measures from the NLP community in our experiments.

7.3. Gold-standard rankings of products

The first challenge when considering the task of ranking products is the question of what conceptual ranking it is that we want to approximate with our generated ranking. Ideally, we would like to have a ranking that reflects how typical users would judge and rank the different products based on their quality. Different users may have different rankings reflecting different uses and preferences, but there should be some sort of consensus ranking. The question is whether such a ranking exists and where we could get it from.

When we look at what has been used in previous work, we can see that in many cases rankings are calculated from the same data that is used for the automatic ranking

methods. For example, Tkachenko and Lauw (2014) create a gold standard of individual textual comparisons that have been manually annotated by crowdsourcing, determine for every pair of products which one is preferred taking into account all comparisons between them and evaluate against this information. Similarly, Ganesan and Zhai (2012) use the average star rating of products in their data. Such “internal” rankings are helpful to see whether the information that is contained in the data can be extracted by automatic methods, but they do not answer the question of whether we can predict a real-world ranking from an external source.

Instead of doing only an intrinsic evaluation, we choose to evaluate against an external gold standard ranking as a more realistic setting of what would be helpful for a user. There are several possibilities that differ in their criteria for ranking products, their availability and their usefulness for different end users. In the following we discuss three possible sources for external gold standards: a sales ranking, an expert ranking, and a crowdsourcing-based annotation of rankings. We mainly discuss them with regard to the following questions (see also Table 7.1 for an overview):

- What are the criteria used for ranking?
- Who can profit from the ranking?
- Is the ranking based on subjective value judgments or objective numbers?
- Who creates the ranking, one person, a group or the crowd?
- Is the user aware of the factors influencing the ranking?
- How hard is it to obtain a ranking for a large number of products?

Sales ranking. One instance of a ranking is the sales ranking of a category of products from an online shop. Figure 7.1 shows a screenshot of the top “Amazon Best Sellers” of the category “DSRL cameras” and Figure 7.2 of the detailed product information from `amazon.com`. The product information contains the sales rank of the product for the relevant product categories (in this case “DSRL cameras” and “Camera & Photo”). The sales rank measures the number of times an item has been sold in a given time period relative to similar products. We could consider this a ranking of products by their economic success. The sales ranking is easy to obtain from a web shop, as it is directly available from structured meta data, and it has been used as an external gold standard ranking in previous work (Zhang et al., 2009, 2013).

The sales ranking can be used by a sales manager or CEO to evaluate the success of a product. As it is based on factual numbers, it is a relatively objective measure of success for a given product, but it does not give an indication about why people buy the product.

Amazon Best Sellers

Our most popular products based on sales. Updated hourly.

Any Department

Camera & Photo

Camera & Photo

Accessories

Binoculars & Scopes

Camcorders

Bags & Cases

Digital Picture Frames

DSLR Cameras

Lenses

Point & Shoot Digital

Cameras

Surveillance Cameras

Best Sellers in DSLR Cameras

1.



Nikon D3300 24.2 MP
CMOS Digital SLR...

★★★★★ (800)

\$396.95

91 used & new from \$327.25

2.



Nikon D3300 DX-format
DSLR Kit w/ 18-...

★★★★★ (800)

\$496.95

44 used & new from \$430.48

3.



Canon EOS Rebel T5 EF-S
18-55mm IS II...

★★★★★ (582)

\$399.00

108 used & new from \$283.00

4.



Nikon D750 FX-format
Digital SLR Came...

★★★★★ (354)

\$1,996.95

47 used & new from \$1,519.00

5.



Canon EOS Rebel T6i
Digital SLR with...

★★★★★ (81)

\$699.00

36 used & new from \$629.00

6.



Canon EOS 6D 20.2 MP
CMOS Digital SLR...

★★★★★ (715)

\$1,399.00

93 used & new from \$1,149.00

7.



Canon EOS 5D Mark III 22.3
MP Full Fr...

★★★★★ (632)

\$2,499.00

79 used & new from \$1,374.00

8.



Nikon D3300 24.2 MP
CMOS Digital SLR...

★★★★★ (800)

\$396.95

28 used & new from \$345.00

9.



Canon EOS Rebel T5
Digital SLR Camera...

★★★★★ (582)

\$449.00

49 used & new from \$356.72

Figure 7.1.: Screenshot of the top products in the category “DSRL cameras” from amazon.com (taken on September 17th, 2015).



Canon EOS-1D X 18.1MP Full Frame CMOS Digital SLR Camera
\$4,599.00 & FREE Shipping. [Details](#) | [In Stock](#). Ships from and sold by Amazon.com. Gift-wrap available.

Product Details

Package Type: **Standard Packaging**

Product Dimensions: 6.2 x 3.3 x 6.5 inches ; 3 pounds

Shipping Weight: 7.2 pounds ([View shipping rates and policies](#))

Domestic Shipping: Item can be shipped within U.S.

International Shipping: This item is not eligible for international shipping. [Learn More](#)

ASIN: B005Y3T1AI

Item model number: 5253B002

Average Customer Review: ★★★★★ (45 customer reviews)

Amazon Best Sellers Rank: #2,123 in Camera & Photo ([See Top 100 in Camera & Photo](#))
#106 in [Camera & Photo](#) > [DSLR Cameras](#)

Manufacturer's warranty can be requested from customer service. [Click here](#) to make a request to customer service.



Date first available at Amazon.com: October 18, 2011

Would you like to [update product info](#), [give feedback on images](#), or [tell us about a lower price?](#)


Figure 7.2.: Screenshot of the product details for Canon EOS 1D X from [amazon.com](#) which contains the sales rank as “Amazon Best Sellers Rank” in the lower part (taken on September 17th, 2015).

This ranking is created by the crowd, the people who buy the product or a competitor product. Apart from the aspects discussed in the reviews, we would also expect the sales rank to be influenced by external factors like advertisements, recommendations of friends, available reviews and even the prices of the same products at competitor shops. The user is typically not fully aware of all factors influencing the rank. If we are able to predict the sales rank from product reviews and extract aspect-specific information, an interesting question is to analyze the impact of the different aspects on the ranking. This information can be used by managers or designers to find out which aspects are the most important for the buying decision of a customer and thus worth investing effort.

Expert ranking. The sales rank measures economic success, which we would assume to be connected to quality, but the connection is not explicit. One source of explicit quality judgments is an expert ranking, which is usually intended to compare different products according to guidelines that reflect typical usages. A common source for such rankings are domain specific magazines or websites with the aim of providing users with a condensed source of information supporting their purchase decision. This ranking is typically created by an individual or a small group of domain experts who use both objectively measurable data (where they determine which value is good or bad) and subjective value judgments. Typically, different aspects are taken into account and evaluated separately. Those aspects might or might not be disclosed, and they may

CANON EOS
1D X Score
Full frame, 18.1 MP, 8.1cm



Retailer amazon.de

Style Body only 2.100,00 €

2.100,00 €

IN STOCK

🛒 Buy Now

	Overview	Competitors	Best prices	Specs	Score	Videos
compared to	DSLRs announced in the last 4 years					
	RANK #1		OUT OF 45		SCORE 100	
			SCORE		WEIGHT	
	DXO MARK SCORES					
Low light performance	🔗 2,786 ISO	<div style="width: 85%; background-color: #8bc34a;"></div>	85.0	x	100.0	= 85.0
Color depth	🔗 23.8 bits	<div style="width: 95%; background-color: #8bc34a;"></div>	95.0	x	25.0	= 23.7
Dynamic range	🔗 11.8 EV	<div style="width: 60%; background-color: #8bc34a;"></div>	60.0	x	25.0	= 15.0
	MISC					
Cross type focus points	🔗 41	<div style="width: 65%; background-color: #8bc34a;"></div>	65.0	x	37.5	= 24.4
Shutter lag	🔗 36 ms	<div style="width: 100%; background-color: #8bc34a;"></div>	100.0	x	37.5	= 37.5
Viewfinder size	🔗 0.76x	<div style="width: 100%; background-color: #8bc34a;"></div>	100.0	x	37.5	= 37.5
Popularity	🔗 3,976	<div style="width: 5%; background-color: #8bc34a;"></div>	5.0	x	20.0	= 1.0
	SCREEN					
Screen resolution	🔗 1,040k dots	<div style="width: 85%; background-color: #8bc34a;"></div>	85.0	x	25.0	= 21.3
Screen size	🔗 8.1cm	<div style="width: 100%; background-color: #8bc34a;"></div>	100.0	x	25.0	= 25.0
Touch screen	🔗 No	<div style="width: 0%; background-color: #8bc34a;"></div>	0.0	x	12.5	= 0.0
	ADVANCED					
Image stabilization	🔗 None	<div style="width: 0%; background-color: #8bc34a;"></div>	0.0	x	25.0	= 0.0
Continuous shooting	🔗 14 fps	<div style="width: 100%; background-color: #8bc34a;"></div>	100.0	x	12.5	= 12.5
	FORM FACTOR					
Viewfinder	🔗 Pentaprism	<div style="width: 100%; background-color: #8bc34a;"></div>	100.0	x	25.0	= 25.0
Lens availability	🔗 165 lenses	<div style="width: 70%; background-color: #8bc34a;"></div>	70.0	x	5.0	= 3.5
	MOVIES					
Movie format	🔗 1080p @ 30fps	<div style="width: 25%; background-color: #8bc34a;"></div>	25.0	x	25.0	= 6.2
			TOTAL		317.6	
			BEST TOTAL		317.6	
			SCORE		100	

Figure 7.3.: Screenshot of the scores including the expert rank for Canon EOS 1D X from snapsort.com (taken on September 17th, 2015).

have different weights on the final score. The list of aspects is usually fixed. Figure 7.3 shows a screenshot of the scores for the different aspects taken into account for a camera at the web page `snapsort.com`, a page dedicated to provide detailed information about cameras and other electronics products. While expert rankings are intuitively the most useful rankings for end users, they are rather costly to produce and so often do not contain as large a number of products as sales rankings.

Expert rankings have been used for evaluation in previous work. Zhang et al. (2009) use a ranking from `smartratings.com`, but only measure the number of products in their top 10% contained in the expert ranking. Tkachenko and Lauw (2014) automatically create a partial expert ranking for three predefined objectively measurable product characteristics. They define good and bad values (e.g., that smaller is better for cameras) and evaluate their system against these aspect-specific expert rankings. In our experiments, we are using a manually created expert ranking that considers many aspects and evaluate against the complete ranking.

Crowdsourced ranking. As only a small group of people is involved in creating an expert rating, the result might be highly subjective, both in the judgments for the individual aspects that are evaluated, and in the selection of these aspects in the first place. The sales ranking attempts to combine opinions from a set of users, but does so in a rather covert way, as the only “opinion” that can be expressed is the decision to buy the product at the specific store. Therefore, we propose that the ideal gold standard product ranking should be based on crowdsourcing without predefining the aspects taken into account. Such a ranking should essentially capture the same information as an expert ranking, with some advantages: (1) the ranking is the work of the crowd instead of a single person or group, so it is more representative of the quality as seen by different people and less likely to be biased towards a specific use, (2) it includes not only measurable criteria, but also subjective judgments, (3) it is not restricted to a fixed set of aspects determined to be important at a given point by an expert, but can include aspects that may only be interesting to a small set of people or very new aspects.

Requesting a full ranking of a list of products from annotators is a cumbersome challenge. Therefore, we propose that such crowdsourcing task should be set up in a learning-to-rank setting, in which annotators are asked to define a preference for a pair of products. The pairwise annotations can then later be used for compiling an inter-subjective ranking as well as a personalized ranking. From such rankings, a personalized preference function can be learned which weights different aspects against each other, even if the user is not aware of these factors. Alternatively, we could ask users to provide reasons

	sales ranking	expert ranking	crowdsourced ranking
Rank criterion?	economic success	product quality	product quality
Target group?	managers	buyers	buyers
Subjective/objective?	objective	both	both
Who?	crowd	experts	crowd
Aware of factors?	no	yes	partly
Availability?	easy to obtain	moderately easy	hard
Size?	large	small	large
Example source?	amazon.com	snapsort.com	–

Table 7.1.: Comparison of the characteristics of different external gold rankings.

for their selections that correspond to the aspects of the product. While it initially requires some effort to set up the crowdsourcing experiment, once that has been done, large amounts of data for different domains could be collected easily. This approach is not performed here, but constitutes relevant future work as we would expect the crowd’s judgments to be the best reflection of actual users’ needs.

Table 7.1 shows an overview of the different possible external gold rankings discussed in this section. While a crowdsourcing ranking would correspond best to our ideal ranking, we currently do not have such a resource. The the sales ranking and the expert ranking present two conceptually very different choices of rankings, which is why we use one example of each for the evaluation of our methods.

7.4. Prediction of product rankings

Our goal is to create a ranked list of products based on review information. We present a method for ranking products according to the information extracted from textual comparisons. We compare this method to two approaches based on subjective phrases and two baselines that take only the review meta data into account.

There are several other possibilities for ranking products that could be explored. Besides the link analysis methods based on product graphs used in previous work, it is conceivable to use a learning-to-rank machine learning framework. In this framework, one training data instance consist of a pair of products represented with some features with the ranking order as the label. The machine learning algorithm will then learn to rank a pair of products that is presented. As for all supervised machine learning tasks, there is a large array of possible feature representations and learning algorithms that could be explored. While these are interesting directions for possible future work, we

concentrate for now on the more straight-forward score-based methods that are presented in the following which are easier to understand and analyze.

7.4.1. Ranking methods

Ranking based on comparisons. As our goal is to ultimately generate a ranked list of products, it is a straight-forward idea to exploit textual comparison expressions:

(7.1) “[*It*]_{E+} has a **better** [*lens*]_A than [*the t3i*]_{E-}”

To extract such comparisons, we employ CSRL (Comparison Semantic Role Labeler, as described in Chapter 5). For each comparison, the system identifies the comparative predicate (“*better*”), the two entities that are involved (“*It*” and “*the t3i*”), which one is preferred (“*It*”), and the compared aspect (“*lens*”).

The entities our system identifies are textual references to products, but they may not always contain the complete product name, so we need to map each mentioned entity to the actual product it refers to. In our mapping process, we associate a mentioned entity to the product name (or names) with the highest cosine similarity on token level. In the example, “*the t3i*” would be associated with the camera “*Canon EOS Rebel T3i*”. If there are several names found, the occurrence is counted for all of them, we do not attempt any disambiguation. Occurrences of the pronoun “*it*” and the expressions “*this*” and “*camera*” are mapped to the product that is the subject of the current review. We treat every entity of the comparison individually, so matches are counted even if only one of the two entities can be mapped to a product.

The final score for a product p is calculated based on the number of times the product occurs as a preferred entity (pref) minus the number of times it occurs as a non-preferred entity (npref) in a comparison:

$$\text{score}_{\text{Csrl}}(p) = \text{pref}(p) - \text{npref}(p) \quad (7.1)$$

The ranking of products is created by sorting according to these scores, where the product with the highest score is at the top of the list. We refer to this method as CSRL. An illustration of the process for some example products can be seen in Figure 7.4.

Ranking based on subjective phrases. The two approaches that we compare to are based on counting words or phrases with a positive and negative polarity.

The first method is a standard term counting approach that is commonly used in sentiment analysis. This approach assigns polarities to words based on a dictionary in

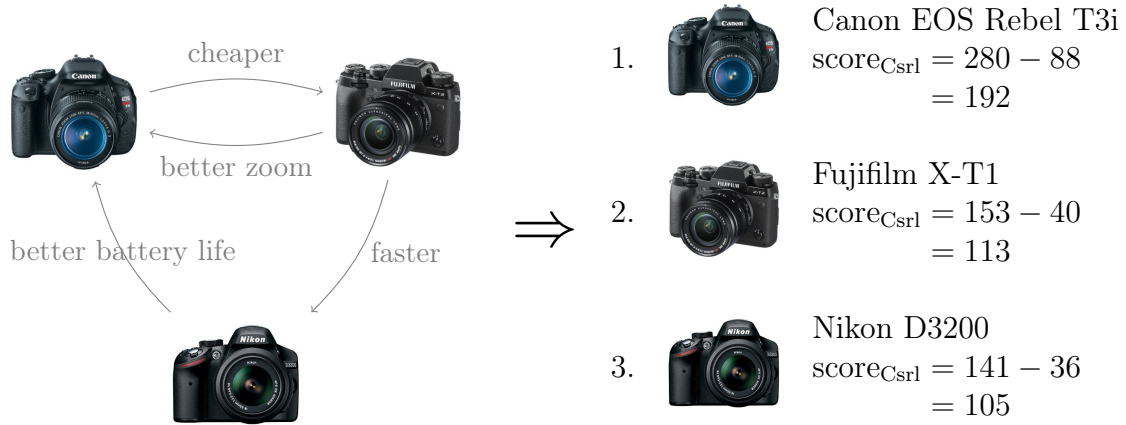


Figure 7.4.: Illustration of ranking based on comparisons.

which the corresponding polarity is explicitly stated. The polarity score for a product p is then calculated as the number of all positive words (pos) in all reviews for this product minus the number of all negative words (neg):

$$\text{score}_{\text{Dict}}(p) = \text{pos}(p) - \text{neg}(p) \quad (7.2)$$

To account for the impact of longer reviews, we normalize the score by the number of tokens in all reviews for the specific product (all):

$$\text{score}_{\text{Dict-Norm}}(p) = \frac{\text{pos}(p) - \text{neg}(p)}{\text{all}(p)} \quad (7.3)$$

The ranked list of products is then created by sorting according to this score. We refer to the two variations of this method as **DICT** and **DICT-NORM**.

This dictionary-based method is easy to implement and to use. However, it might not take into account context specific polarity expressions, e.g., negation. As a second method, we therefore opt for machine learning detection of subjective phrases with their polarities in context. Specifically we use Roman Klinger's tool **JFSA**² (Joint Fine-Grained Sentiment Analysis Tool, Klinger and Cimiano (2013)). **JFSA** uses Conditional Random Fields to jointly detect subjective phrases and their targets. Calculating the product score $\text{score}_{\text{JFSA}}(p)$ is performed analogous to the dictionary-based approach by counting all positive (pos) and negative (neg) subjective phrases identified by the system in all reviews for this product. Ranking and normalization are the same as for **DICT**. We refer to the two variations of this method as **JFSA** and **JFSA-NORM**.

²<https://bitbucket.org/rklinger/jfsa>

Baselines. We use two baselines that do not take the textual information of a review into account. The first method, STARS, sorts products by their average star rating (from one to five stars, as assigned by the author of a review). The average star rating is calculated from all reviews of a product. The second baseline, NUMREVIEWS, sorts the products by the number of reviews a product has received (from none to many). The intuition is that products which are sold more often gather more reviews, so we expect this to be a strong baseline for predicting the sales rank.

7.4.2. Aspect-specific rankings

Two of our methods, JFSA and CSRL, recognize aspects of products together with a subjective phrase or comparison, respectively. Besides creating one ranking that is a combined measure of all aspects of the product, we have the option of using only evaluations regarding specific aspects to calculate the scores for the products. This results in an aspect-specific ranking. In the aspect-specific version of a method, subjective phrases or entity mentions are only counted towards the score of a product if there is a token overlap between the mentioned aspect and a textual variation of the target aspect.

For CSRL, we can directly use the aspect (or aspects) the system associated with a comparative predicate. For JFSA, aspects and subjective phrases are identified separately and the relation detection has low performance. Thus, we associate each subjective phrase with the identified aspect that is closest. One aspect can be the closest aspect for several subjective phrases and is counted for each of them.

One challenge of aspect-specific rankings is the large number of different aspect phrases that are used. It is quite common to use different words or phrases to refer to the same aspect. To illustrate the scope of the problem, in our comparison data the 1800 aspect annotations correspond to over 800 different phrases. Consider the phrases “*image*”, “*picture*” and “*pics*”. In the camera domain, all of them are used to refer to the same aspect *picture*. For our aspect-specific rankings, we would like to group the different phrases together into one aspect. In some cases, grouping is an issue of identifying synonyms (e.g., “*image*” and “*picture*”), orthographic variations and abbreviations (“*picture*” and “*pics*”). But additionally, aspects form a fine-grained hierarchy, for example “*battery life*” and “*battery charge time*” are different aspects of the aspect *battery*. Most of the time, the set of all extracted aspects will be too fine-grained to be useful for an end-user, so it is necessary to group them on some level of the hierarchy.

Aspect grouping is a difficult problem and, as far as we know no grouping or hierarchy of aspects is publicly available for the camera domain. Thus, for our initial experiments

<i>aspect name</i>	<i>matching phrases</i>
pictures	images, image, photos, photo, pictures, picture, shots, shot, shooting, image quality, pics, picture quality, results
lens	lens, lenses, lense
price	price, cost, costs, \$, cheap, expensive
battery	battery, batteries, charger, charge, battery life
flash	flash
features	features, feature, settings, options, functions
size	size, weight
video	video, videos, film, films, movie, movies, record, records, recording
zoom	zoom
performance	work, works, worked, performance, perform, performs, performed, speed
shutter	shutter
screen	screen, lcd, lcd screen
autofocus	autofocus, af

Table 7.2.: List of aspect phrases used for aspect-specific rankings.

on aspect-specific rankings, we use short manually compiled lists of textual variations for the most frequent aspects in our data set. These lists are shown in Table 7.2. The lists have been created by looking at the most frequent aspect phrases found by the two systems on our data and grouping them. We then chose the groups that seemed the most relevant as aspects to consider for our initial experiments. As the more infrequent phrases have not been considered, the lists include only a tiny percentage of the aspect phrases actually found in the data.

7.5. Experiments

7.5.1. Data and experimental setup

For our evaluation, we use camera reviews retrieved from `amazon.com`³. As our first gold ranking, we extract the Amazon sales rank from the product descriptions (“Amazon Best Sellers Rank” in the “Camera & Photo” category) as retrieved between April 14th and 18th, 2015. We include only products for which a rank is provided in our final data

³The reviews were crawled by Roman Klinger with the search term “camera” and “camera” in conjunction with “fuji”, “fujifilm”, “canon”, “panasonic”, “olympus”, “nikon”, “sigma”, “hasselblad”, “leica”, “pentax”, “rollei”, “samsung”, “sony”, “olympus”.

set. The resulting list contains 920 products with a total of 71,409 reviews. Product names are extracted from the title of the description page and shortened to the first six tokens to remove additional descriptions of the product like “*with 28x Optical Zoom and 3.0-Inch LCD (black)*”.

As a second external gold ranking, we use the expert ranking from `snapsort.com`. Snapsort is a service that collects detailed information about cameras and offers comparisons between them. Their product score incorporates aspects from technical specifications (e.g., shutter, viewfinder size, whether image stabilization is available), as well as popularity (e.g., how often the camera has been viewed on the website, number of lenses available for the camera). From the top 150 products in the Amazon sales ranking, 56 are found on Snapsort. We use the rank in the category “best overall” of “all digital cameras announced in the last 48 month” as retrieved on June 12th, 2015.⁴

The JFSA system is trained on the polarity annotations in the camera data set by J. Kessler et al. (2010). CSRL is trained on our IMS camera data described in Chapter 4. As we are only interested in comparisons where there is a ranking between the entities, we train only on comparisons annotated as types “ranked” and “superlative”. To reduce confusions of the preferred and non-preferred entity, we also exclude all comparisons where the second entity is preferred over the first (i.e., direction is “negative”). The final training data consists of 1019 sentences with 1248 predicates.

For the methods DICT and DICT-NORM, we test two different sources of opinion words, the General Inquirer dictionary GI (Stone et al., 1966)⁵ and the MPQA subjectivity clues (Wilson et al., 2005)⁶.

To measure the correlation of the rankings generated by our different methods with the gold ranking, we calculate Spearman’s rank correlation coefficient ρ (Spearman, 1904) and test for significance with the Steiger test (Steiger, 1980). A correlation coefficient of 0 indicates no correlation, with a value of 1 the rankings would be the same, and when one ranking is the inverse of the other the correlation would be -1 .

7.5.2. Results and discussion

As described above, we use two different rankings for evaluation: The `amazon.com` sales ranking, which contains 920 products, and the expert ranking from `snapsort.com`, which contains 56 products. These two rankings are conceptually different, with economic

⁴The full list of products with their names and ranks is available from <http://aclweb.org/anthology/attachments/W/W15/W15-2908.Attachment.tgz>.

⁵3518 entries; 1775 positive, 1743 negative using the categories from (Choi and Cardie, 2008).

⁶6456 entries; 2304 positive, 4152 negative.

Method	Amazon	Snapsort
STARS	-0.027	0.436*
NUMREVIEWS	0.331*	0.095
DICT-NORM (GI)	0.125*	-0.148
DICT (GI)	0.219*	0.426*
DICT-NORM (MPQA)	0.142*	-0.145
DICT (MPQA)	0.222*	0.441*
JFSA-NORM	0.151*	-0.230
JFSA	0.234*	0.404*
CSRL	0.183*	0.511*

Table 7.3.: Results (Spearman’s ρ) of all methods for predicting the Amazon sales ranking and the Snapsort quality ranking. Significance over random is marked with * ($p < 0.05$). The best baseline and the best text-based method are marked in bold.

success as the main factor for the sales ranking and product quality for the expert ranking. This conceptual difference is reflected in the fact that there is no correlation between the two rankings ($\rho = -0.04$).

Results with all aspects. Table 7.3 shows the results for the baselines and the text-based methods on both gold rankings when using all aspects. For all systems, there is a pronounced difference between the results for the two gold rankings.

The best result on Amazon (significantly outperforming all other methods) is achieved by counting the reviews ($\rho = 0.33$, NUMREVIEWS). For Snapsort, however, NUMREVIEWS leads to a correlation of only $\rho = 0.1$. One factor that explains this difference in performance is the fact that in case of Amazon the reviews and the ranking come from the same source. While it is unclear whether the popularity of a product leads to many reviews or a high number of reviews leads to higher sales, there is a good chance that the two things are correlated. Though “popularity” is one aspect that influences the Snapsort rating, it is not as prominent.

The performance of the STARS baseline is not significantly different from random for Amazon. This is partly due to unreliable ratings when products have only very few reviews (less than 10) which happens for many products with a 5.0 star rating. Having few reviews is less of a problem for the products in the Snapsort ranking. On the Snapsort data, the STARS baseline achieves the second-best result. It is understandable that star ratings are more closely aligned with a ranking by product quality than with a sales ranking. The sales ranking may be influenced by many factors that may not be discussed

Aspect	#	ρ	σ
performance	637	0.301	0.009
video	600	0.278	0.013
size	513	0.218	0.017
pictures	790	0.213	0.003
battery	541	0.208	0.012
price	625	0.198	0.008
zoom	514	0.196	0.013
shutter	410	0.191	0.016
features	629	0.190	0.009
autofocus	403	0.175	0.013
screen	501	0.136	0.012
lens	457	0.099	0.012
flash	591	0.093	0.011

Table 7.4.: Results (Spearman’s ρ and standard deviation σ) of JFSA for predicting the aspect-specific Amazon sales ranking. The number of products for which at least one evaluation of the target aspect is found is shown in column #.

in reviews nor reflected in the star rating, e.g., advertisements, recommendations from friends, or competitor’s prices. Both our baseline’s results are similar to the results reported by Zhang et al. (2013) who use the same baselines on a presumably similar data set of Amazon reviews of digital cameras evaluated against the Amazon sales rank (they report a correlation of 0.109 for STARS, 0.331 for NUMREVIEWS).

The results for all non-normalized ranking-methods based on subjective phrases (DICT (MPQA), DICT (GI), JFSA) are similar to each other. When we compare the resulting rankings of the systems among each other, all of the correlation scores are above 0.95 (not shown in the table). JFSA achieves the best performance on the Amazon data, while DICT (MPQA) is best on the Snapsort data. In all cases, normalization of the polarity scores hurts performance. The non-normalized rankings are not correlated with the normalized rankings of the same system (ρ is between -0.19 and 0.02 , not shown in the table). In general, correlations are much higher on the Snapsort data.

On the Amazon data, the ranking achieved with CSRL is mediocre in comparison to the other methods. On the Snapsort ranking however, CSRL leads to the best result of all experiments with $\rho = 0.51$, which is very encouraging. CSRL suffers more from data sparsity than the other methods. The highest number of subjective phrases for a product found by JFSA is over 9000, while the highest number of comparisons that mention a given product is 662 for CSRL. For 105 products no mention at all is found by CSRL, while this happens only to 6 products for JFSA.

Aspect-specific results. In comparison to using all information extracted from reviews to generate a ranking, the aspect-specific results allow for an understanding of the impact of each aspect on the gold ranking. Aspect-specific rankings for important aspects are highly correlated with the gold ranking, while those for completely irrelevant aspects have a correlation near zero. Due to data sparsity, for a substantial amount of products no mentions of a specific aspect are found, so they receive a final ranking score of zero. To eliminate the resulting artificial inflation of ρ while enabling a comparison between methods with different numbers of scored products, we add the zero-scoring products to our ranking in random order and average over 100 random rankings for the zero-scoring products. The results for JFSA on the Amazon sales ranking are shown in Table 7.4. We omit the results for CSRL and the results on Snapsort which are all close to random.

From our small selection of aspects, the aspect *performance* contributes most to approximating the sales ranking ($\rho = 0.30$) followed by *video* ($\rho = 0.28$). Both results outperform the target-agnostic ranking of JFSA (the difference is significant for *performance*). *performance* is the second most frequent aspect, after *pictures*.

7.6. Summary

In this chapter we have presented the task of producing a ranked list of products and discussed several real-world products rankings. We demonstrated how we can use the information extracted from textual comparison expressions to create a ranking of products. We compared our method to baselines that use review meta data and methods on the basis of subjective phrases. We evaluated against two conceptually different external gold rankings, the sales ranking from `amazon.com` and an expert ranking from `snapsort.com`. For the expert ranking, our comparison-based method outperformed all other methods and showed the highest performance of all experiments. We also presented experiments on how aspect-specific rankings can be used to measure the impact of one specific aspect on the overall ranking. While in general the performance of all systems in terms of correlation scores is rather low, the task of approximating a real world ranking looks promising and encouraging for further research.

8. Conclusions and outlook

8.1. Summary of contributions

The objective of this thesis was to explore how robust structurally informed models for sentiment analysis can be developed, addressing challenges that arise from polarity reversers and comparisons, two examples of structurally complex verbalizations of opinions. The contribution made to this end are summarized in the following.

Research Question A: *How can structural linguistic context information be used for the reliable detection of complex verbalizations of opinions?*

In Chapter 3 we presented a machine learning classifier to distinguish whether a given sentiment word in context is consistent or inconsistent with its prior polarity from a sentiment dictionary, i.e., determine whether a polarity reverser is present in the context around the sentiment word. We investigated different ways of representing relevant sentence context around the sentiment word. We demonstrated that for our classifier using dependency path-based features for representing sentence context improves over the window-based context used in previous work (Ikeda et al., 2008).

In Chapter 5 we presented CSRL (Comparison Semantic Role Labeler), a machine learning system trained for the task of detecting the individual components of comparisons: the anchor or predicate of the comparison, the entities that are compared, which aspect they are compared in, and which entity is preferred. We experimented with three different types of context: Sentence context, context from knowledge about the task and the domain, and context from the annotation design of the data used for training our machine learning system. Similar to our experiments for polarity reversers, we used dependency tree information for sentence context and compared to a window-based context used in previous work (Jindal and Liu, 2006b). For comparative predicates, adding both types of context proved to be beneficial, but for arguments structured context clearly outperformed window-based context. To include context from the task and domain, we used generalization techniques to overcome sparsity issues, added information about

possible types of comparisons, and included sentiment polarity information. These experiments yielded mixed results. We demonstrated that annotation design decisions, especially those concerning the linguistic anchoring of multiword predicates, have a considerable impact on the overall classification performance and present a detailed analysis of the results of systematically varying these annotations.

Research Question B: *How can the structure of complex verbalizations of opinions be exploited in order to automatically annotate training examples by using semi-supervised methods?*

In Chapter 6 we showed that we can exploit the structure of comparison to expand a small seed set of labeled comparisons with the semi-supervised method of structural alignment, proposed for Semantic Role Labeling by Fürstenaу and Lapata (2009, 2012). This method starts with a small seed set of labeled seed sentences, finds sentences in a large corpus of unlabeled data that are similar to the seed sentences, and projects the labels onto them. By adding the newly annotated sentences to the training data for our system, the initial small manually annotated seed set of sentences is expanded to create a set large enough for efficient machine learning without additional manual annotation effort. Capturing the structure of comparisons is the key step in this process to ensure that the found sentences are similar enough so that the projection of labels onto them produces correctly labeled new training examples, but also different enough to include linguistic variations that enable the classifier to learn new information. We proposed adaptations to two steps of structural alignment as a way of tailoring the approach to the structural characteristic of comparisons: the similarity measure and the method of extracting candidates for comparative arguments. Our experiments showed that these adaptations improve the overall performance of CSRL trained on the expanded data compared to a system without these adaptations.

Research Question C: *How can information found in textual comparisons be used to support a process of choice?*

In Chapter 7 we worked on the task of producing a ranked list of products that complements the isolated prediction of ratings to support the user’s process of choice. We presented a method to create a ranking over a set of products by aggregating all individual comparisons found in the data. Specifically, we calculate a score based on the number of times a product is mentioned as the preferred or non-preferred entity in a comparison. We compared these comparison-based rankings to other methods of

creating a ranking that are based on subjective phrases and review meta data. We evaluated on two external rankings, a sales ranking, which reflects the number of times an item has been sold in a given time period relative to similar products, and an expert ranking, for which a domain expert compares different products and ranks them by their quality. While our system performed average when compared to the sales ranking, it achieved the best result for the expert ranking.

Accompanying this thesis, a dedicated gold standard of manually annotated comparisons has been created and made available to the research community. The corpus is based on the English camera review data by Branavan et al. (2009) and contains 2200 sentences and 2700 comparisons with detailed annotations, which makes it the largest such resource currently available to our knowledge. The resource is described in more detail in Chapter 4. The data is available from <http://hdl.handle.net/11022/1007-0000-0000-8E72-0>. Other resources that have been created and made available to the community include a list of manually annotated polarity reversing constructions, polarity consistency annotations on word-level, review sentences automatically annotated with polarity (all Chapter 3), and product rankings for cameras from the amazon.com sales ranking and the snapsort.com expert ranking (Chapter 7).

8.2. Possible directions for future work

Polarity reversers. Our approach for polarity classification in Chapter 3 is based on a dictionary of sentiment words. The approach detects many words that do not express sentiment in a given context. Identifying and discarding such non-subjective uses of sentiment words would improve the overall classification results. Also, as not all sentiment expressions contain sentiment words, the performance that can be achieved with a dictionary-based approach in general is limited, so alternative approaches should be considered. We have presented first steps towards automatically extracting polarity reversing constructions (PRCs) from sentences annotated with polarity. The results of this extraction are rather noisy and can be improved in many ways by filtering the results, e.g., to filter out short candidates where longer candidates are already identified as good reversers. To get sufficient training data for the extraction of PRCs, we have automatically assigned polarity annotations to sentences extracted from semi-structured reviews which are easily available in large quantities. The quality and coverage of this automatic annotation could be improved in many ways, e.g., by addressing the issue of different words that are used to refer to the same aspect of a product. We could also use clustering methods on the reviews instead of the direct matching of keywords.

Comparison detection. In Chapter 5 we have worked on the task of comparison detection and investigated structural features based on the output of a dependency parser. As we work on user generated content (product reviews) and use a parser that was trained on standard news texts, the resulting parse trees are rather noisy. Improvements in POS tagging and parsing this type of data could improve our results as well. Possible future work that builds more directly on our experiments consists of including more sophisticated features to model context, addressing the issue of sentiment relevance, and addressing annotation design issues besides systematically introduced multiword predicates. We have addressed three types of context in this work, but there are many more forms of context that could be investigated, e.g., discourse context or context about the products that are reviewed.

Training data expansion. In Chapter 6 we have presented two main adaptations of structural alignment for our task of comparison detection, but many others are conceivable. Two main issues are false positive and false negative predicates found by the expansion. False positive predicates are introduced by not detecting non-subjective usage of comparative words. This issue could be addressed with methods from subjectivity analysis to filter out non-comparative usages. False negative predicates are generated through other predicates besides the identified one being present in an expansion sentence. The issue could be addressed by simplifying sentences, extracting only the relevant sentence parts or preselecting only short and simple sentences for expansion. On a related note, not all annotated predicates may be ideally suited to be seeds for the expansion process. We could select only “typical” predicates as seeds for the expansion, i.e., predicates that are similar to a number of other predicates already in the training data. Similarly, we could try to discard sentences of bad quality from the set of seed or expansion sentences, e.g., sentences that are very long, have non-standard punctuation, or parse-trees with a very low confidence score. Finally, we assume a fixed value of k sentences to be extracted for each seed sentence. It may be beneficial to use a variable value for k that depends on the seed sentence and the found possible expansion sentences.

Ranking. In Chapter 7 we have presented a method to create a ranking over a set of products by aggregating all individual product comparisons found in the data. As there has not been a lot of research in this area, there are a lot of open issues that can be addressed. We have touched on the discussion of how to set up a crowdsourcing-based annotation of rankings that may better approximate an ideal ranking that could be used as a reliable gold standard for evaluation. Possible future work on the methods side

includes the formulation of the problem as a learning-to-rank task. This would allow us to combine the different measures discussed in the chapter. For aspect-specific rankings, an automatic way to group the large number of different aspect phrases that are used in reviews would be desirable, potentially by using hierarchical clustering.

General Remarks. The area of sentiment analysis is a rather new field of research and as such many basic challenges still need to be addressed. In our opinion, the most important open issue is subjectivity or rather sentiment relevance, which we have found to impact our results on many occasions. Principled solutions on this topic could potentially have a large benefit for the sentiment analysis community.

A. Detailed results for Chapter 3

ADV<not_ADV	ADJ<V<if_PR	N<break_V
ADV<V<V>not_ADV	ADJ<N<V<would_0	N<slow_V
ADJ<V>not_ADV	ADJ<V<O>not_ADV	ADV<V<V>poor_ADJ
ADV<V<would_0	N<PR<ADJ>not_ADV	N<O<N>bad_ADJ
ADV<O>not_ADV	N<without_PR	N<problem_N
ADV<ADJ>not_ADV	N>not_ADV	ADV<PR>N>poor_ADJ
N>no_DT	N<V<would_0	N<ADV<problem_N
N>low_ADJ	ADJ>no_DT	ADV<N>poor_ADJ
ADJ>not_ADV	ADJ<N<PR<lack_N	V<PR<delay_V
ADJ<N>not_ADV	N<V>never_ADV	ADV<PR<N>poor_ADJ
ADJ<N>no_DT	V>not_ADV	N<PR<slow_ADJ
ADV<V>not_ADV	ADJ<N<PR<less_ADJ	N<N<problem_N
ADJ<N<without_PR	N<PR<lack_N	ADV<N<V>poor_ADJ
ADV<ADJ<V>not_ADV	V<O<hard_ADJ	N<PR<V<problem_N
ADJ<V<V>not_ADV	V<without_PR	V<PR<delay_N
ADJ>only_ADV	V>never_ADV	ADV<PR>V>poor_ADJ
ADJ<not_ADV	V<O<leave_V	N<N>N>slow_ADJ
ADJ>less_ADV	N<V>bad_ADJ	N<V>bad_N
ADJ<V<would_0	V<V<O<leave_V	V<delay_N
V<V>not_ADV	N<PR<problem_N	N<O<break_V
ADJ<V<could_0	N<PR<N>bad_ADJ	ADV<too_ADV
ADJ<V<V<would_0	N<slow_ADJ	V<O>not_ADV
V<would_0	V<O<O<leave_V	
ADJ<N<V>not_ADV	N<N<V>bad_ADJ	

Table A.1.: List of manually annotated polarity reversing constructions. This list is also available from <http://hdl.handle.net/11022/1007-0000-0000-8E6F-5>.

Representation	Scoring method	1	2	3	4	5	6
LEMMAS	f_{rel}	0.10	0.06	0.06	0.00	0.00	0.00
	MI	0.10	0.13	0.14	0.13	0.11	0.13
	MI+	0.10	0.10	0.06	0.01	0.00	0.00
SIMPLE PATHS	f_{rel}	0.16	0.11	0.07	0.06	0.01	0.01
	MI	0.13	0.16	0.16	0.14	0.14	0.13
	MI+	0.17	0.21	0.19	0.14	0.13	0.13
ABSTRACTED PATHS	f_{rel}	0.14	0.17	0.19	0.20	0.20	0.20
	MI	0.13	0.20	0.19	0.19	0.19	0.19
	MI+	0.17	0.24	0.27	0.29	0.29	0.29

(a) Basic results varying representation and scoring method.

Filter	1	2	3	4	5	6	
BL / ABSTRACT PATHS, MI+	0.17	0.24	0.27	0.29	0.29	0.29	
SUBJSTRENGTH	0.09	0.17	0.23	0.21	0.21	0.21	
POSIGNORE	0.17	0.29	0.29	0.31	0.31	0.31	
INTENSIFIER	0.16	0.23	0.26	0.29	0.29	0.29	
SINGLETONS	0.17	0.27	0.27	0.30	0.30	0.30	
ASPECTS	50	0.16	0.24	0.26	0.29	0.29	0.29
	70	0.16	0.24	0.26	0.29	0.29	0.29
	100	0.16	0.24	0.26	0.29	0.29	0.29
	150	0.16	0.26	0.26	0.29	0.29	0.29
	200	0.16	0.26	0.26	0.27	0.27	0.27
DOMAIN	500	0.09	0.17	0.20	0.23	0.23	0.23
	1000	0.11	0.20	0.24	0.24	0.24	0.24
	2000	0.14	0.23	0.21	0.23	0.23	0.23
	3000	0.14	0.24	0.23	0.24	0.24	0.24
	4000	0.16	0.20	0.23	0.24	0.24	0.24
	5000	0.14	0.19	0.20	0.21	0.21	0.21
	6000	0.16	0.20	0.20	0.21	0.21	0.21
	7000	0.16	0.21	0.23	0.24	0.24	0.24
	8000	0.17	0.23	0.27	0.27	0.27	0.27
	9000	0.17	0.24	0.27	0.29	0.29	0.29
10000	0.17	0.24	0.27	0.29	0.29	0.29	
POSIGNORE+ INTENSIFIER	0.16	0.27	0.29	0.30	0.30	0.30	
POSIGNORE + SINGLETONS	0.17	0.29	0.30	0.31	0.31	0.31	
POSIGNORE + DOMAIN 1000	0.11	0.26	0.26	0.29	0.29	0.29	
POSIGNORE + ASPECTS 100	0.16	0.27	0.30	0.31	0.31	0.31	

(b) Results with different filters on top of the best basic method (ABSTRACT PATHS, MI+).

Table A.2.: Detailed results of PRC extraction with path lengths between $k = 1$ and $k = 6$ (Precision). Bold numbers denote the best result for each k and scoring method.

		A	class inconsistent			class consistent			F_m
			P	R	F	P	R	F	
MPQA, HL	standard vot. BASELINE	91.2	0.0	0.0	0.0	91.2	100.0	95.4	47.7*
	negation vot. WORDS	92.2	56.8	45.5	50.5	94.9	96.7	95.8	73.1*
	negation vot. PRC-GOLD	93.5	64.2	58.0	60.9	96.0	96.9	96.4	78.7
	negation vot. PRC-SYSTEM	91.1	49.3	52.1	50.7	95.4	94.8	95.1	72.9*
MPQA, DL	standard vot. BASELINE	90.7	0.0	0.0	0.0	90.7	100.0	95.1	47.6*
	negation vot. WORDS	91.3	54.4	40.6	46.5	94.1	96.5	95.3	70.9*
	negation vot. PRC-GOLD	91.8	56.4	50.5	53.3	95.0	96.0	95.5	74.4
	negation vot. PRC-SYSTEM	90.0	46.0	46.7	46.4	94.5	94.4	94.5	70.4*
MPQA, HL	standard vot. BASELINE	91.2	0.0	0.0	0.0	91.2	100.0	95.4	47.7*
	classifier vot. BoW	82.1	25.9	55.6	35.3	95.2	84.7	89.6	62.5
	classifier vot. BoC	86.2	30.3	44.8	36.2	94.4	90.1	92.2	64.2
	classifier vot. BoPRC-GOLD	92.6	59.7	50.6	54.2	95.3	96.6	96.0	75.2*
	classifier vot. BoPRC-SYSTEM	91.8	53.4	51.8	52.5	95.4	95.6	95.5	74.0*
MPQA, DL	standard vot. BASELINE	90.7	0.0	0.0	0.0	90.7	100.0	95.1	47.6*
	classifier vot. BoW	80.3	24.0	51.5	32.7	94.4	83.3	88.4	60.6*
	classifier vot. BoC	86.7	34.5	47.5	40.0	94.4	90.8	92.5	66.3
	classifier vot. BoPRC-GOLD	92.8	67.0	44.1	53.1	94.5	97.8	96.1	74.6*
	classifier vot. BoPRC-SYSTEM	92.8	64.9	48.8	55.6	94.9	97.3	96.1	75.9*

Table A.3.: Detailed results of word-level consistency classification (Accuracy, Precision, Recall, F_1 score, macro F_1 (F_m)). The best result for each data set is bolded. F_m results are marked with * if the difference to negation vot. PRC-GOLD resp. classifier vot. BoC is statistically significant at $p < .05$.

	WORDS	PRC GOLD	PRC SYSTEM		WORDS	PRC GOLD	PRC SYSTEM		
BASELINE	0.0000	0.0000	0.0000	BASELINE	0.0000	0.0000	0.0000		
WORDS		0.0004	0.4472	WORDS		0.0097	0.3913		
PRC-GOLD			0.0000	PRC-GOLD			0.0007		
(a) Negation vot., MPQA, HL				(b) Negation vot., MPQA, DL					
	BoW	BoC	BoPRC GOLD	BoPRC SYSTEM		BoW	BoC	BoPRC GOLD	BoPRC SYSTEM
BASELINE	0.0000	0.0000	0.0000	0.0000	BASELINE	0.0000	0.0000	0.0000	0.0000
BoW		0.1544	0.0000	0.0000	BoW		0.0000	0.0000	0.0000
BoC			0.0000	0.0000	BoC			0.0000	0.0000
BoPRC-GOLD				0.1469	BoPRC-GOLD				0.1509
(c) Classifier vot., MPQA, HL					(d) Classifier vot., MPQA, DL				

Table A.4.: Statistical significances (p -values) for word-level consistency classification. Values are marked in bold if $p < 0.05$.

			A	class positive			class negative			F_m
				P	R	F	P	R	F	
MPQA, HL	standard vot.	BASELINE	76.1	76.5	91.7	83.4	74.5	46.4	57.2	70.3*
	negation vot.	WORDS	79.0	78.4	93.7	85.4	80.9	51.0	62.6	74.0*
	negation vot.	PRC-GOLD	80.2	79.6	93.8	86.1	82.1	54.2	65.3	75.7
	negation vot.	PRC-SYSTEM	78.8	78.4	93.2	85.2	79.9	51.2	62.4	73.8*
MPQA, DL	standard vot.	BASELINE	73.0	75.3	87.7	81.0	65.2	44.6	53.0	67.0*
	negation vot.	WORDS	75.0	77.1	88.2	82.3	68.5	49.5	57.5	69.9*
	negation vot.	PRC-GOLD	76.4	78.3	88.8	83.2	70.8	52.5	60.3	71.8
	negation vot.	PRC-SYSTEM	75.1	76.6	89.6	82.6	70.2	47.3	56.5	69.6*
GI, HL	standard vot.	BASELINE	70.8	72.6	86.8	79.0	65.2	42.9	51.8	65.4*
	negation vot.	WORDS	72.9	74.2	87.9	80.5	69.0	47.0	55.9	68.2
	negation vot.	PRC-GOLD	74.2	74.8	89.5	81.5	72.3	47.5	57.4	69.4
	negation vot.	PRC-SYSTEM	72.7	73.9	88.3	80.4	69.2	45.7	55.0	67.7*
GI, DL	standard vot.	BASELINE	67.1	70.9	82.6	76.3	55.4	38.9	45.7	61.0*
	negation vot.	WORDS	69.0	72.3	83.8	77.7	59.1	42.2	49.2	63.5*
	negation vot.	PRC-GOLD	70.8	73.7	84.9	78.9	62.5	45.4	52.6	65.8
	negation vot.	PRC-SYSTEM	69.1	72.1	84.7	77.9	59.6	40.8	48.4	63.2*
MPQA, HL	standard vot.	BASELINE	76.1	76.5	91.7	83.4	74.5	46.4	57.2	70.3*
	classifier vot.	BOW	78.0	80.4	87.8	84.0	71.9	59.3	65.0	74.5*
	classifier vot.	BOC	80.5	81.8	90.5	85.9	77.3	61.5	68.5	77.2
	classifier vot.	BOPRC-GOLD	80.9	81.6	91.6	86.3	79.1	60.5	68.6	77.4
	classifier vot.	BOPRC-SYSTEM	81.3	81.5	92.5	86.7	80.9	60.0	68.9	77.8
MPQA, DL	standard vot.	BASELINE	73.0	75.3	87.7	81.0	65.2	44.6	53.0	67.0*
	classifier vot.	BOW	74.5	78.3	84.9	81.5	65.2	54.5	59.4	70.4*
	classifier vot.	BOC	77.5	79.8	88.1	83.8	71.3	57.0	63.4	73.6
	classifier vot.	BOPRC-GOLD	76.6	79.4	87.0	83.0	69.2	56.5	62.2	72.6
	classifier vot.	BOPRC-SYSTEM	77.2	79.2	88.6	83.6	71.4	55.2	62.3	72.9
GI, HL	standard vot.	BASELINE	70.8	72.6	86.8	79.0	65.2	42.9	51.8	65.4*
	classifier vot.	BOW	72.9	75.4	84.9	79.9	66.5	52.0	58.4	69.1
	classifier vot.	BOC	74.5	77.2	84.7	80.8	68.1	56.6	61.8	71.3
	classifier vot.	BOPRC-GOLD	74.4	75.7	87.8	81.3	70.7	51.1	59.3	70.3
	classifier vot.	BOPRC-SYSTEM	74.1	75.4	88.1	81.2	70.7	49.8	58.4	69.8
GI, DL	standard vot.	BASELINE	67.1	70.9	82.6	76.3	55.4	38.9	45.7	61.0*
	classifier vot.	BOW	69.2	72.7	83.3	77.7	59.2	43.5	50.1	63.8*
	classifier vot.	BOC	72.2	76.5	82.1	79.2	62.8	54.5	58.3	68.8
	classifier vot.	BOPRC-GOLD	70.0	73.4	83.8	78.3	60.8	45.1	51.8	65.0*
	classifier vot.	BOPRC-SYSTEM	70.4	73.3	84.8	78.7	61.8	44.3	51.6	65.1*

Table A.5.: Detailed results of sentence-level polarity classification (**A**ccuracy, **P**recision, **R**ecall, **F**₁ score, macro F_1 (**F**_m)). Averaged over 10 random runs of 10-fold cross-validation. Bold numbers denote the best result for each data set and sentiment word list. F_m results are marked with * if the difference to negation vot. with PRC-GOLD resp. classifier vot. with BOC is statistically significant at $p < .05$.

	WORDS	PRC GOLD	PRC SYSTEM
BASELINE	0.0000	0.0000	0.0002
WORDS		0.0128	0.4251
PRC-GOLD			0.0079

(a) Negation vot., MPQA, HL

	WORDS	PRC GOLD	PRC SYSTEM
BASELINE	0.0001	0.0000	0.0033
WORDS		0.0094	0.3700
PRC-GOLD			0.0055

(b) Negation vot., MPQA, DL

	WORDS	PRC GOLD	PRC SYSTEM
BASELINE	0.0004	0.0000	0.0073
WORDS		0.0596	0.2797
PRC-GOLD			0.0155

(c) Negation vot., GI, HL

	WORDS	PRC GOLD	PRC SYSTEM
BASELINE	0.0004	0.0000	0.0164
WORDS		0.0030	0.3799
PRC-GOLD			0.0012

(d) Negation vot., GI, DL

	BoW	BoC	BoPRC GOLD	BoPRC SYSTEM
BASELINE	0.0027	0.0000	0.0000	0.0000
BoW		0.0375	0.0209	0.0118
BoC			0.3349	0.2665
BoPRC-GOLD				0.2690

(e) Classifier vot., MPQA, HL

	BoW	BoC	BoPRC GOLD	BoPRC SYSTEM
BASELINE	0.0059	0.0000	0.0000	0.0000
BoW		0.0097	0.0434	0.0289
BoC			0.1877	0.2734
BoPRC-GOLD				0.2937

(f) Classifier vot., MPQA, DL

	BoW	BoC	BoPRC GOLD	BoPRC SYSTEM
BASELINE	0.0122	0.0000	0.0000	0.0000
BoW		0.0955	0.2208	0.2812
BoC			0.2140	0.1180
BoPRC-GOLD				0.2300

(g) Classifier vot., GI, HL

	BoW	BoC	BoPRC GOLD	BoPRC SYSTEM
BASELINE	0.0298	0.0000	0.0000	0.0000
BoW		0.0021	0.2211	0.2045
BoC			0.0037	0.0054
BoPRC-GOLD				0.2617

(h) Classifier vot., GI, DL

Table A.6.: Statistical significances (p -values) for sentence-level polarity classification. Values are marked in bold if $p < 0.05$.

B. Detailed results for Chapter 5

		Predicate ident.			Argument ident.			Argument class.		
		<i>P</i>	<i>R</i>	<i>F</i>	<i>P</i>	<i>R</i>	<i>F</i>	P_μ	R_μ	F_μ
IMS	BL POS	84.4	47.4	60.7*	–	–	–	–	–	–
	BASELINE	67.8	68.7	68.2*	44.0	46.8	45.3*	33.4	35.6	34.5*
	MINIMAL	83.9	75.7	79.6*	57.4	23.7	33.6*	55.1	22.8	32.2*
	WINDOW	87.0	77.3	81.9	68.4	47.7	56.2*	63.7	44.5	52.4*
	SYNTAX	85.7	78.1	81.7	80.0	59.3	68.1	74.9	55.5	63.8
	WINDOW+SYNTAX	87.2	76.2	81.3	64.9	49.9	56.4*	60.3	46.4	52.4*
J-A	BL POS	62.5	34.7	44.6*	–	–	–	–	–	–
	BASELINE	51.8	56.5	54.1*	50.0	49.8	49.9*	43.7	43.5	43.6*
	MINIMAL	69.3	60.1	64.4*	53.1	9.6	16.2*	52.6	9.5	16.0*
	WINDOW	75.2	57.5	65.2	61.5	28.1	38.6*	58.5	26.7	36.7*
	SYNTAX	74.6	59.5	66.2	70.7	42.1	52.8	66.4	39.5	49.5
	WINDOW+SYNTAX	75.2	56.1	64.3*	59.0	31.8	41.3*	55.8	30.1	39.1*
J-C	BL POS	66.6	38.2	48.5*	–	–	–	–	–	–
	BASELINE	53.1	62.8	57.5*	49.7	47.2	48.4	44.0	41.8	42.9*
	MINIMAL	71.2	57.6	63.7*	58.3	14.0	22.5*	57.3	13.7	22.2*
	WINDOW	77.0	58.3	66.3	63.3	27.6	38.5*	60.8	26.6	37.0*
	SYNTAX	74.2	59.2	65.9	66.9	37.3	47.9	63.7	35.5	45.6
	WINDOW+SYNTAX	76.7	57.9	66.0*	58.1	29.9	39.5*	56.2	29.0	38.2*
J&L	BL POS	74.1	53.0	61.8*	–	–	–	–	–	–
	BASELINE	61.3	80.1	69.4	40.1	45.7	42.7*	28.7	32.7	30.6*
	MINIMAL	73.7	68.3	70.9	57.5	13.3	21.6*	52.7	12.2	19.8*
	WINDOW	78.6	67.9	72.9	65.9	33.5	44.5*	60.4	30.7	40.7*
	SYNTAX	76.4	66.8	71.3	69.5	46.4	55.7	63.1	42.1	50.5
	WINDOW+SYNTAX	78.7	66.3	72.0	62.8	37.0	46.6*	57.2	33.7	42.4*

Table B.1.: Detailed results of the experiments on sentence context (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Numbers for arguments are on gold predicates. The best result in each experiment group, data set and task is marked in bold. F_1 score results are marked with * if the difference to SYNTAX is significant at $p < .05$.

	KEY	MIN	WIN	SYN	W+S
POS	0.0000	0.0000	0.0000	0.0000	0.0000
KEYWORDS		0.0000	0.0000	0.0000	0.0000
MINIMAL			0.0000	0.0000	0.0002
WINDOW				0.3777	0.0254
SYNTAX					0.1253

(a) Predicate identification, IMS

	KEY	MIN	WIN	SYN	W+S
POS	0.0000	0.0000	0.0000	0.0000	0.0000
KEYWORDS		0.0000	0.0000	0.0000	0.0000
MINIMAL			0.1765	0.0160	0.4687
WINDOW				0.1265	0.0618
SYNTAX					0.0053

(b) Predicate identification, J-A

	KEY	MIN	WIN	SYN	W+S
POS	0.0000	0.0000	0.0000	0.0000	0.0000
KEYWORDS		0.0000	0.0000	0.0000	0.0000
MINIMAL			0.0326	0.0485	0.0566
WINDOW				0.3728	0.3810
SYNTAX					0.4533

(c) Predicate identification, J-C

	KEY	MIN	WIN	SYN	W+S
POS	0.0000	0.0000	0.0000	0.0000	0.0000
KEYWORDS		0.0952	0.0063	0.0983	0.0341
MINIMAL			0.0342	0.3843	0.1845
WINDOW				0.0570	0.1200
SYNTAX					0.2017

(d) Predicate identification, J&L

Table B.2.: Statistical significances (p -values) for experiments on sentence context for predicate identification. Values are marked in bold if $p < 0.05$.

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0000	0.0000	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.2573
SYN				0.0000

(a) Argument identification, IMS

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0019	0.0000	0.0000	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.4622
SYN				0.0000

(b) Argument classification, IMS

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0000	0.0014	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0000
SYN				0.0000

(c) Argument identification, J-A

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0000	0.0000	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0000
SYN				0.0000

(d) Argument classification, J-A

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0000	0.3523	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0688
SYN				0.0000

(e) Argument identification, J-C

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0001	0.0104	0.0005
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0301
SYN				0.0000

(f) Argument classification, J-C

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.1114	0.0045	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0055
SYN				0.0000

(g) Argument identification, J&L

	MINIMAL	WINDOW	SYNTAX	W+S
BL	0.0000	0.0000	0.0000	0.0000
MIN		0.0000	0.0000	0.0000
WIN			0.0000	0.0331
SYN				0.0000

(h) Argument classification, J&L

Table B.3.: Statistical significances (p -values) for experiments on sentence context for argument identification and classification. Values are marked in bold if $p < 0.05$.

		Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
IMS	SYNTAX	85.7	78.1	81.7	80.0	59.3	68.1	74.9	55.5	63.8
	NER	85.6	78.5	81.9	79.6	60.4	68.7	74.8	56.7	64.5
	CLUSTER	85.8	78.0	81.7	79.5	59.9	68.3	74.4	56.0	63.9
	TYPE	–	–	–	75.0	64.1	69.1*	70.1	59.9	64.6*
	POLARITY	85.8	78.1	81.8	80.5	59.1	68.1	75.4	55.3	63.8
J-A	SYNTAX	74.6	59.5	66.2	70.7	42.1	52.8	66.4	39.5	49.5
	NER	74.3	59.4	66.0	70.7	43.2	53.6	66.6	40.7	50.6
	CLUSTER	74.8	58.8	65.8	69.5	43.0	53.1	65.2	40.3	49.8
	TYPE	–	–	–	68.8	43.4	53.2*	64.4	40.6	49.8
	POLARITY	74.5	59.2	66.0	70.8	41.9	52.6	66.4	39.3	49.3
J-C	SYNTAX	74.2	59.2	65.9	66.9	37.3	47.9	63.7	35.5	45.6
	NER	74.3	59.8	66.3	67.2	41.0	51.0	64.6	39.4	49.0
	CLUSTER	75.0	59.3	66.3	64.8	38.7	48.5	62.1	37.1	46.4
	TYPE	–	–	–	64.8	40.4	49.7*	61.7	38.4	47.3*
	POLARITY	74.4	58.4	65.4	68.8	37.3	48.4	65.3	35.5	46.0
J&L	SYNTAX	76.4	66.8	71.3	69.5	46.4	55.7	63.1	42.1	50.5
	NER	76.5	67.3	71.6	70.3	48.8	57.6	64.4	44.7	52.8
	CLUSTER	77.2	67.3	71.9	68.4	47.0	55.7	62.5	43.0	50.9
	TYPE	–	–	–	64.7	53.7	58.7*	58.5	48.6	53.1*
	POLARITY	76.4	67.2	71.5	69.8	46.1	55.5	63.5	41.9	50.5

Table B.4.: Detailed results of the experiments with task and domain context (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Numbers for arguments are on gold predicates. The best result in each experiment group, data set and task is marked in bold. F_1 score results are marked with * if the difference to SYNTAX is significant at $p < .05$.

	CLUSTER			TYPE		POLARITY		
	Pred.	A-ident.	A-class.	A-ident.	A-class.	Pred.	A-ident.	A-class.
IMS	0.5012	0.1364	0.2580	0.0000	0.0001	0.3512	0.3149	0.2087
J-A	0.1521	0.2071	0.2906	0.0424	0.2012	0.1510	0.1710	0.1426
J-C	0.2818	0.1452	0.0647	0.0000	0.0000	0.1164	0.0445	0.0818
J&L	0.1332	0.4680	0.2663	0.0000	0.0000	0.2984	0.3032	0.4942

Table B.5.: Statistical significances (p -values) relative to SYNTAX for experiments on context from task and domain. Values are marked in bold if $p < 0.05$.

		Predicate ident.				Argument ident.				Argument class.			
		P	R	F	ΔF	P	R	F	ΔF	P_μ	R_μ	F_μ	ΔF_μ
IMS	FUNCTION	85.7	78.1	81.7		80.0	59.3	68.1		74.9	55.5	63.8	
	CONTENT	85.5	75.2	80.0	-1.7	79.4	58.8	67.6	-0.5	74.0	54.8	62.9	-0.9
J-A	FUNCTION	74.6	59.5	66.2		70.7	42.1	52.8		66.4	39.5	49.5	
	CONTENT	74.6	53.2	62.1	-4.1	70.3	42.0	52.6	-0.2	66.6	39.8	49.8	+0.3
J-C	FUNCTION	74.2	59.2	65.9		66.9	37.3	47.9		63.7	35.5	45.6	
	CONTENT	75.7	54.8	63.6	-2.3	66.7	37.4	47.9	-0.0	63.0	35.3	45.2	-0.4
J&L	FUNCTION	76.4	66.8	71.3		69.5	46.4	55.7		63.1	42.1	50.5	
	CONTENT	75.2	60.9	67.3	-4.0	69.7	45.3	54.9	-0.8	62.4	40.6	49.2	-1.3

(a) Results for FUNCTION predicates vs. CONTENT predicates.

		Argument classification			
		P_μ	R_μ	F_μ	ΔF_μ
IMS	UNIFIED	74.9	55.5	63.8	
	SCALE	74.4	55.1	63.3	-0.5
	SURFACE	70.1	51.9	59.6	-4.2
	PREFERENCE	64.7	47.9	55.0	-8.8
J-A	UNIFIED	66.4	39.5	49.5	
	SCALE	65.9	39.2	49.2	-0.3
	SURFACE	60.9	36.3	45.4	-4.1
	PREFERENCE	58.5	34.8	43.6	-5.9
J-C	UNIFIED	63.7	35.5	45.6	
	SCALE	63.2	35.3	45.3	-0.3
	SURFACE	57.8	32.2	41.4	-4.2
	PREFERENCE	56.4	31.5	40.4	-5.2
J&L	UNIFIED	63.1	42.1	50.5	
	SCALE	62.4	41.7	50.0	-0.5
	SURFACE	60.0	40.1	48.1	-2.4
	PREFERENCE	-	-	-	

(b) Argument classification results for changing argument labels. Predicate and argument identification are not affected by these changes.

Table B.6.: Detailed results of the experiments on annotation design context (Precision, Recall, F_1 score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Numbers for arguments are on gold predicates. Bold numbers denote the best result in each data set and task. ΔF denotes the absolute differences in F_1 score to the FUNCTION / UNIFIED setting (equivalent to SYNTAX in Table B.1), as we cannot calculate statistical significance.

C. Detailed results for Chapter 6

		Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
$k=1$	BASELINE	100.0	6.1	11.5	100.0	5.6	10.6	52.4	2.9	5.6
	VS	100.0	6.1	11.5	96.7	5.3	10.1	55.7	3.1	5.8
	WINDOW	100.0	6.1	11.5	100.0	5.5	10.5	61.8	3.4	6.5
	TREEWINDOW	100.0	6.1	11.5	96.8	5.4	10.2	62.1	3.5	6.6
	DEP	100.0	6.1	11.5	100.0	5.5	10.4	51.6	2.8	5.4
	POSITION	100.0	6.1	11.5	99.2	5.5	10.4	64.2	3.6	6.7
	LEVEL	100.0	6.1	11.5	98.4	5.5	10.4	66.9	3.7	7.1
	PATH	100.0	6.1	11.5	100.0	5.5	10.3	64.5	3.5	6.7
$k=2$	BASELINE	100.0	11.7	21.0	98.8	10.7	19.3	49.6	5.4	9.7
	VS	100.0	11.7	21.0	97.9	10.4	18.8	52.1	5.5	10.0
	WINDOW	100.0	11.7	21.0	99.6	10.7	19.3	57.6	6.2	11.2
	TREEWINDOW	100.0	11.7	21.0	98.3	10.5	18.9	60.2	6.4	11.6
	DEP	100.0	11.7	21.0	99.1	10.5	18.9	55.6	5.9	10.6
	POSITION	100.0	11.7	21.0	99.2	10.6	19.1	60.8	6.5	11.7
	LEVEL	100.0	11.7	21.0	98.8	10.7	19.3	65.4	7.1	12.8
	PATH	100.0	11.7	21.0	98.7	10.5	19.0	65.3	6.9	12.6
$k=3$	BASELINE	100.0	16.8	28.7	98.8	15.2	26.4	56.4	8.7	15.1
	VS	100.0	16.8	28.7	97.6	14.8	25.7	47.9	7.3	12.6
	WINDOW	100.0	16.8	28.7	99.4	15.3	26.6	52.0	8.0	13.9
	TREEWINDOW	100.0	16.8	28.7	98.5	14.9	25.9	60.3	9.1	15.8
	DEP	100.0	16.8	28.7	98.5	14.8	25.8	57.8	8.7	15.1
	POSITION	100.0	16.8	28.7	99.1	15.0	26.0	60.3	9.1	15.8
	LEVEL	100.0	16.8	28.7	98.2	15.1	26.3	64.0	9.9	17.1
	PATH	100.0	16.8	28.7	98.8	14.8	25.8	66.7	10.0	17.4

Table C.1.: Detailed results on development set for LABELED arguments on both sides with different argument similarities and different k (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Bold numbers denote the best result for each setting and task (predicate identification, argument identification, argument classification).

		Predicate ident.			Argument ident.			Argument class.		
		<i>P</i>	<i>R</i>	<i>F</i>	<i>P</i>	<i>R</i>	<i>F</i>	P_μ	R_μ	F_μ
<i>k</i> = 5	BASELINE	100.0	26.4	41.8	99.0	23.4	37.9	55.5	13.1	21.2
	VS	100.0	26.4	41.8	98.4	22.9	37.2	48.8	11.4	18.4
	WINDOW	100.0	26.4	41.8	98.9	23.3	37.7	50.0	11.8	19.1
	TREEWINDOW	100.0	26.4	41.8	98.2	22.7	36.8	59.2	13.7	22.2
	DEP	100.0	26.4	41.8	98.4	22.8	37.0	56.9	13.2	21.4
	POSITION	100.0	26.4	41.8	98.6	22.8	37.0	61.1	14.1	22.9
	LEVEL	100.0	26.4	41.8	98.7	23.3	37.7	64.5	15.2	24.7
	PATH	100.0	26.4	41.8	98.4	22.4	36.5	66.3	15.1	24.6
<i>k</i> = 10	BASELINE	100.0	46.1	63.1	99.0	40.8	57.8	53.1	21.9	31.0
	VS	100.0	46.1	63.1	98.6	40.0	56.9	45.9	18.7	26.5
	WINDOW	100.0	46.1	63.1	98.9	40.3	57.2	46.5	18.9	26.9
	TREEWINDOW	100.0	46.1	63.1	98.8	39.6	56.5	54.7	21.9	31.3
	DEP	100.0	46.1	63.1	99.0	39.2	56.2	54.0	21.4	30.7
	POSITION	100.0	46.1	63.1	99.0	39.3	56.3	58.3	23.2	33.2
	LEVEL	100.0	46.1	63.1	98.7	40.4	57.3	56.7	23.2	32.9
	PATH	100.0	46.1	63.1	99.1	38.7	55.7	63.3	24.8	35.6
<i>k</i> = optimal	BASELINE	100.0	98.2	99.1	99.1	77.0	86.6	50.6	39.3	44.3
	VS	100.0	98.2	99.1	99.0	76.9	86.6	38.9	30.2	34.0
	WINDOW	100.0	98.2	99.1	99.1	77.0	86.6	43.8	34.0	38.3
	TREEWINDOW	100.0	98.2	99.1	99.0	76.9	86.5	50.0	38.8	43.7
	DEP	100.0	98.2	99.1	99.3	77.1	86.8	50.4	39.2	44.1
	POSITION	100.0	98.2	99.1	99.0	76.9	86.6	50.7	39.4	44.3
	LEVEL	100.0	98.2	99.1	99.1	77.0	86.6	49.3	38.3	43.1
	PATH	100.0	98.2	99.1	99.4	77.2	86.9	55.6	43.2	48.6

Table C.2.: Detailed results on development set for LABELED arguments on both sides with different argument similarities and different *k* (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Bold numbers denote the best result for each setting and task (predicate identification, argument identification, argument classification).

		Predicate ident.			Argument ident.			Argument class.		
		<i>P</i>	<i>R</i>	<i>F</i>	<i>P</i>	<i>R</i>	<i>F</i>	P_μ	R_μ	F_μ
FL	BASELINE	69.3	34.5	46.0	34.6	9.2	14.5	9.8	2.6	4.1
	VS	68.2	33.9	45.3	40.5	11.4	17.7	14.3	4.0	6.3
	WINDOW	68.0	33.8	45.1	37.6	9.5	15.1	16.8	4.2	6.8
	TREEWINDOW	67.4	33.5	44.8	38.5	11.2	17.3	21.4	6.2	9.6
	DEP	67.8	33.7	45.0	43.1	11.4	18.0	23.9	6.3	10.0
	POSITION	66.9	33.2	44.4	41.2	11.0	17.4	24.4	6.5	10.3
	LEVEL	66.7	33.2	44.3	35.9	10.8	16.6	18.7	5.6	8.7
	PATH	64.6	32.1	42.9	41.9	11.3	17.8	27.2	7.3	11.5
KK	BASELINE	45.5	25.1	32.3	11.9	5.5	7.5	6.0	2.7	3.8
	VS	48.1	26.5	34.2	21.5	10.1	13.7	10.8	5.1	6.9
	WINDOW	45.1	24.9	32.1	16.8	7.7	10.6	9.7	4.5	6.1
	TREEWINDOW	54.2	29.9	38.5	22.0	10.3	14.0	13.7	6.4	8.7
	DEP	51.5	28.4	36.6	21.9	10.2	13.9	13.6	6.3	8.7
	POSITION	45.5	25.1	32.3	15.7	7.3	10.0	11.4	5.3	7.3
	LEVEL	47.2	26.0	33.6	19.4	9.0	12.3	12.9	6.0	8.2
	PATH	50.6	27.9	36.0	28.2	13.1	17.9	19.9	9.2	12.6
FL-sym	BASELINE	67.8	33.7	45.0	36.0	12.9	19.0	13.7	4.9	7.2
	VS	66.7	33.2	44.3	38.3	13.8	20.3	14.6	5.3	7.8
	WINDOW	66.7	33.2	44.3	35.5	12.7	18.7	12.9	4.6	6.8
	TREEWINDOW	65.5	32.5	43.5	34.7	12.3	18.2	18.4	6.5	9.6
	DEP	67.4	33.5	44.8	40.6	14.0	20.8	22.7	7.8	11.6
	POSITION	66.3	33.0	44.1	37.5	12.9	19.2	22.3	7.7	11.4
	LEVEL	66.3	33.0	44.1	33.0	12.1	17.7	16.8	6.2	9.0
	PATH	64.6	32.1	42.9	38.0	13.0	19.4	25.2	8.7	12.9
KK-sym	BASELINE	47.3	26.0	33.6	11.1	5.8	7.6	6.3	3.3	4.3
	VS	52.8	29.1	37.6	24.4	12.9	16.8	13.9	7.3	9.6
	WINDOW	49.4	27.2	35.1	20.1	10.6	13.9	12.1	6.4	8.4
	TREEWINDOW	54.2	29.9	38.5	21.7	11.5	15.0	14.6	7.7	10.1
	DEP	49.8	27.5	35.4	23.5	12.4	16.3	14.7	7.8	10.2
	POSITION	47.2	26.0	33.6	17.7	9.4	12.3	12.1	6.4	8.4
	LEVEL	53.0	29.3	37.7	21.6	11.4	15.0	14.7	7.8	10.2
	PATH	53.2	29.4	37.9	29.7	15.7	20.6	20.9	11.0	14.4

Table C.3.: Detailed results on development set for argument candidate extraction with different argument similarities for $k = 10$ (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Bold numbers denote the best result for each argument creation method (symmetric FL, KK, and asymmetric FL-SYM, KK-SYM) and task (predicate identification, argument identification, argument classification).

		Predicate ident.			Argument ident.			Argument class.		
		<i>P</i>	<i>R</i>	<i>F</i>	<i>P</i>	<i>R</i>	<i>F</i>	P_μ	R_μ	F_μ
FL	BASELINE	69.7	34.7	46.3	44.4	12.8	19.9	28.2	8.2	12.7
	VS	68.2	33.9	45.3	45.5	13.1	20.4	28.3	8.2	12.7
	WINDOW	68.2	33.9	45.3	43.4	12.6	19.5	28.3	8.2	12.7
	TREEWINDOW	65.5	32.5	43.5	42.1	12.0	18.7	27.0	7.7	12.0
	DEP	66.5	33.1	44.2	42.8	12.1	18.9	28.7	8.1	12.6
KK	BASELINE	50.4	27.8	35.8	27.2	12.9	17.5	19.8	9.4	12.7
	VS	50.4	27.8	35.8	27.4	13.0	17.6	19.4	9.2	12.5
	WINDOW	50.8	28.0	36.1	27.9	13.2	18.0	20.2	9.6	13.0
	TREEWINDOW	48.7	26.9	34.6	27.8	13.2	17.9	20.7	9.8	13.3
	DEP	50.2	27.7	35.7	28.2	13.4	18.1	19.9	9.5	12.8
FL-sym	BASELINE	67.8	33.7	45.0	40.1	14.5	21.3	25.8	9.3	13.7
	VS	67.4	33.5	44.8	41.1	14.7	21.6	26.0	9.3	13.7
	WINDOW	66.7	33.2	44.3	38.9	13.9	20.5	25.7	9.2	13.5
	TREEWINDOW	63.4	31.5	42.1	37.1	13.1	19.3	23.9	8.4	12.5
	DEP	65.9	32.8	43.8	39.3	13.8	20.4	26.2	9.2	13.6
KK-sym	BASELINE	54.9	30.3	39.1	32.4	17.1	22.4	23.9	12.6	16.5
	VS	57.0	31.5	40.5	32.5	17.2	22.5	23.9	12.6	16.5
	WINDOW	54.2	29.9	38.5	31.9	16.8	22.0	23.9	12.6	16.5
	TREEWINDOW	51.7	28.6	36.8	30.3	16.0	21.0	22.6	11.9	15.6
	DEP	52.5	29.0	37.3	31.4	16.6	21.7	23.3	12.3	16.1

Table C.4.: Detailed results on development set for different predicate similarities for $k = 10$ (**P**recision, **R**ecall, **F**₁ score, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes). Bold numbers denote the best result for each argument creation method (symmetric FL, KK, and asymmetric FL-SYM, KK-SYM) and task (predicate identification, argument identification, argument classification).

Explanation for all following tables:

Detailed results on test set for all versions of structural alignment (FL-FLAT, KK-FLAT, FL-CONTEXT, KK-CONTEXT, FL-CONTEXT-PREDS, KK-CONTEXT-PREDS). The system is trained on data with different number of k expansion sentences added per seed sentence. Results are given for different percentages of seed data d that are used (10%, 25%, 50%, 100%). For predicate and argument identification, **P**recision, **R**ecall, and **F**₁ score are given, for argument classification P_μ , R_μ , F_μ are micro-averaged over all classes.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
FL-FLAT $k = 1$	10	85.3	34.9	49.6	66.6	10.9	18.7	41.2	6.7	11.6
	25	85.4	45.4	59.3	70.5	17.2	27.7	57.6	14.0	22.6
	50	85.7	52.3	65.0	70.5	23.0	34.7	60.5	19.7	29.8
	100	86.6	58.0	69.4	72.2	28.3	40.7	62.5	24.5	35.2
FL-FLAT $k = 2$	10	87.0	38.4	53.3	68.8	11.8	20.2	37.5	6.5	11.0
	25	86.1	42.3	56.7	71.9	15.6	25.6	58.9	12.7	21.0
	50	87.3	48.6	62.4	72.8	21.3	33.0	59.8	17.5	27.1
	100	85.8	54.0	66.3	72.7	26.7	39.1	61.9	22.8	33.3
FL-FLAT $k = 3$	10	86.9	38.1	53.0	71.1	11.9	20.4	39.4	6.6	11.3
	25	86.7	41.2	55.9	71.5	15.1	24.9	57.0	12.0	19.8
	50	88.2	49.8	63.6	72.7	21.9	33.7	58.6	17.7	27.1
	100	86.9	53.9	66.5	73.1	26.3	38.7	61.8	22.2	32.7
FL-FLAT $k = 4$	10	86.9	38.0	52.9	69.3	11.6	19.8	37.2	6.2	10.6
	25	86.2	41.8	56.3	71.7	15.4	25.3	56.4	12.1	19.9
	50	87.8	48.7	62.7	71.6	21.2	32.7	57.7	17.1	26.4
	100	87.5	52.5	65.6	72.6	25.7	38.0	61.0	21.6	31.9
FL-FLAT $k = 5$	10	86.8	40.1	54.9	70.6	12.4	21.1	38.0	6.7	11.4
	25	85.7	43.9	58.1	70.4	16.3	26.5	55.5	12.9	20.9
	50	87.4	48.5	62.4	70.8	21.3	32.8	57.2	17.2	26.5
	100	87.4	50.6	64.1	72.2	24.8	36.9	60.1	20.6	30.7
FL-FLAT $k = 6$	10	86.6	39.6	54.4	69.3	12.3	20.9	36.4	6.5	11.0
	25	85.6	42.3	56.7	69.1	15.8	25.7	54.5	12.4	20.2
	50	87.5	48.3	62.2	69.3	21.4	32.7	55.4	17.1	26.1
	100	87.3	49.0	62.8	71.5	24.5	36.4	59.8	20.5	30.5
FL-FLAT $k = 7$	10	86.8	40.3	55.1	68.3	12.7	21.5	35.5	6.6	11.2
	25	86.0	42.2	56.6	68.1	16.1	26.1	52.9	12.5	20.3
	50	87.6	46.4	60.7	69.8	20.6	31.9	55.9	16.5	25.5
	100	85.6	47.6	61.2	70.3	23.9	35.7	58.1	19.8	29.5
FL-FLAT $k = 8$	10	87.5	38.9	53.8	68.7	12.4	21.1	37.2	6.7	11.4
	25	86.1	41.7	56.2	68.5	16.1	26.1	52.3	12.3	19.9
	50	87.5	45.0	59.4	69.3	20.1	31.2	55.4	16.1	24.9
	100	85.9	47.9	61.5	69.6	24.3	36.0	57.8	20.2	29.9
FL-FLAT $k = 9$	10	87.2	36.1	51.1	68.0	11.4	19.5	36.0	6.0	10.3
	25	86.0	41.8	56.3	68.0	16.3	26.3	52.1	12.5	20.2
	50	86.9	46.2	60.3	68.6	20.4	31.4	54.7	16.3	25.1
	100	85.5	48.4	61.8	69.3	24.3	36.0	57.0	20.0	29.6
FL-FLAT $k = 10$	10	87.3	35.4	50.4	69.4	11.5	19.7	35.1	5.8	10.0
	25	85.9	41.1	55.6	67.0	15.8	25.6	50.8	12.0	19.4
	50	87.4	45.4	59.7	69.3	20.3	31.4	55.4	16.3	25.1
	100	85.9	47.9	61.5	69.3	23.8	35.4	56.6	19.4	28.9

Table C.5.: Detailed results on test set for system FL-FLAT.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
FL-CONTEXT $k = 1$	10	83.8	38.4	52.7	63.9	10.8	18.5	42.1	7.1	12.2
	25	84.3	44.6	58.3	70.6	15.6	25.6	58.2	12.9	21.1
	50	86.5	51.0	64.2	72.5	22.7	34.6	61.2	19.2	29.2
	100	86.1	56.2	68.0	71.7	27.4	39.7	61.4	23.5	34.0
FL-CONTEXT $k = 2$	10	85.5	37.0	51.7	66.0	10.6	18.2	43.4	7.0	12.0
	25	84.9	44.6	58.5	71.0	17.1	27.5	59.0	14.2	22.8
	50	86.0	48.1	61.7	70.9	20.8	32.2	59.7	17.5	27.1
	100	86.0	55.1	67.1	71.8	26.7	38.9	62.0	23.1	33.6
FL-CONTEXT $k = 3$	10	86.3	36.5	51.4	69.3	11.4	19.6	43.5	7.2	12.3
	25	84.6	45.4	59.1	70.8	17.7	28.3	58.1	14.5	23.2
	50	86.1	49.3	62.7	70.8	21.3	32.7	59.1	17.8	27.3
	100	86.1	56.2	68.1	71.6	28.2	40.4	61.3	24.1	34.6
FL-CONTEXT $k = 4$	10	85.4	37.4	52.0	69.5	11.7	20.0	43.1	7.3	12.4
	25	84.0	45.1	58.7	68.9	17.8	28.3	56.4	14.5	23.1
	50	86.1	49.3	62.7	70.9	20.7	32.1	59.4	17.4	26.9
	100	85.8	54.8	66.9	69.8	26.7	38.6	60.2	23.0	33.3
FL-CONTEXT $k = 5$	10	85.9	39.6	54.2	69.3	13.2	22.2	43.3	8.3	13.9
	25	83.9	42.7	56.6	66.7	16.8	26.9	53.1	13.4	21.4
	50	85.0	47.1	60.6	69.0	20.5	31.6	56.5	16.8	25.9
	100	84.3	51.9	64.3	67.8	25.6	37.2	57.7	21.8	31.7
FL-CONTEXT $k = 6$	10	86.4	37.8	52.6	68.5	12.2	20.8	42.6	7.6	12.9
	25	83.8	41.0	55.1	66.9	16.2	26.1	52.2	12.6	20.3
	50	86.1	46.2	60.2	69.3	20.1	31.2	56.6	16.4	25.5
	100	84.2	49.7	62.5	66.1	23.8	35.0	55.5	20.0	29.4
FL-CONTEXT $k = 7$	10	86.2	36.9	51.7	67.3	12.3	20.8	40.7	7.5	12.6
	25	84.1	41.2	55.3	66.7	16.4	26.3	51.8	12.7	20.4
	50	85.4	45.2	59.1	67.7	19.8	30.7	55.0	16.1	24.9
	100	84.7	50.3	63.1	66.4	24.7	36.0	55.8	20.8	30.3
FL-CONTEXT $k = 8$	10	86.6	33.9	48.7	67.9	11.6	19.8	42.3	7.2	12.4
	25	83.1	38.9	52.9	65.2	15.5	25.0	50.3	11.9	19.3
	50	84.9	43.5	57.5	67.9	19.6	30.4	55.2	15.9	24.7
	100	85.8	48.9	62.3	66.8	24.3	35.6	55.4	20.2	29.6
FL-CONTEXT $k = 9$	10	85.8	34.4	49.1	68.1	12.2	20.7	44.0	7.9	13.4
	25	82.8	37.3	51.4	63.9	14.9	24.1	49.7	11.6	18.8
	50	84.2	42.4	56.4	65.7	18.8	29.3	53.3	15.3	23.7
	100	84.9	47.8	61.1	65.6	23.8	34.9	54.9	19.9	29.2
FL-CONTEXT $k = 10$	10	86.6	34.5	49.4	68.1	12.3	20.9	44.2	8.0	13.5
	25	83.1	37.6	51.8	63.4	14.9	24.2	48.3	11.4	18.4
	50	84.3	43.2	57.1	65.2	19.2	29.7	53.4	15.8	24.3
	100	85.4	47.8	61.3	66.3	24.3	35.5	55.3	20.3	29.7

Table C.6.: Detailed results on test set for system FL-CONTEXT.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
FL-CONTEXT-PREDS $k = 1$	10	84.8	43.3	57.4	66.2	12.5	21.0	47.4	8.9	15.0
	25	84.9	46.7	60.3	67.2	16.7	26.7	54.3	13.5	21.6
	50	85.6	55.6	67.5	70.3	23.8	35.6	60.0	20.3	30.4
	100	85.5	61.3	71.4	71.8	29.6	42.0	62.3	25.7	36.4
FL-CONTEXT-PREDS $k = 2$	10	85.9	42.9	57.2	67.7	13.5	22.5	43.5	8.7	14.4
	25	85.0	48.7	62.0	70.4	18.6	29.5	56.9	15.1	23.8
	50	85.4	55.4	67.2	70.7	24.4	36.3	60.8	21.0	31.2
	100	85.4	58.7	69.5	72.9	29.2	41.7	62.5	25.0	35.7
FL-CONTEXT-PREDS $k = 3$	10	85.8	41.7	56.1	67.9	13.2	22.0	40.9	7.9	13.3
	25	85.2	46.5	60.2	72.2	18.2	29.1	58.1	14.6	23.4
	50	86.6	53.9	66.4	70.5	24.1	35.9	61.1	20.8	31.1
	100	85.7	56.7	68.3	72.4	28.0	40.3	62.0	24.0	34.6
FL-CONTEXT-PREDS $k = 4$	10	85.2	43.2	57.4	65.8	13.2	22.0	37.4	7.5	12.5
	25	84.9	46.3	59.9	72.4	18.1	28.9	58.6	14.6	23.4
	50	86.8	52.6	65.5	71.5	23.7	35.5	61.2	20.3	30.5
	100	86.3	55.4	67.5	72.0	27.7	40.0	60.8	23.4	33.8
FL-CONTEXT-PREDS $k = 5$	10	85.7	42.5	56.8	66.9	13.5	22.5	35.5	7.2	11.9
	25	85.5	43.9	58.0	70.5	17.5	28.0	56.5	14.0	22.5
	50	86.1	51.2	64.2	69.3	23.5	35.1	59.3	20.1	30.0
	100	86.1	55.1	67.2	70.7	27.3	39.4	59.5	23.0	33.2
FL-CONTEXT-PREDS $k = 6$	10	85.7	40.8	55.3	66.2	12.7	21.3	34.6	6.7	11.2
	25	85.6	42.3	56.6	69.3	16.8	27.0	54.9	13.3	21.4
	50	86.3	50.4	63.6	68.4	22.9	34.3	57.1	19.1	28.7
	100	86.5	55.1	67.3	70.7	27.3	39.4	59.6	23.0	33.2
FL-CONTEXT-PREDS $k = 7$	10	86.3	40.2	54.8	65.6	12.9	21.6	34.0	6.7	11.2
	25	85.4	41.8	56.1	68.8	16.6	26.7	54.2	13.1	21.1
	50	86.7	49.9	63.4	67.5	22.7	34.0	56.3	18.9	28.3
	100	87.3	54.9	67.4	70.4	27.2	39.2	58.4	22.5	32.5
FL-CONTEXT-PREDS $k = 8$	10	87.3	43.2	57.8	67.9	14.5	24.0	36.4	7.8	12.8
	25	86.1	41.4	55.9	67.4	16.2	26.2	52.5	12.6	20.4
	50	86.6	49.9	63.3	68.2	22.9	34.2	57.3	19.2	28.8
	100	87.7	55.1	67.7	70.2	27.2	39.2	58.1	22.5	32.5
FL-CONTEXT-PREDS $k = 9$	10	87.3	40.7	55.5	68.8	13.9	23.1	37.4	7.5	12.6
	25	86.6	41.5	56.1	68.8	16.8	27.0	54.4	13.3	21.3
	50	87.2	49.1	62.9	68.1	22.6	33.9	56.9	18.8	28.3
	100	87.3	54.6	67.2	69.0	26.6	38.4	57.0	21.9	31.7
FL-CONTEXT-PREDS $k = 10$	10	87.9	40.3	55.3	67.3	13.8	22.9	37.2	7.6	12.6
	25	86.9	42.7	57.3	67.8	17.2	27.5	54.0	13.7	21.9
	50	87.3	50.1	63.7	67.3	22.9	34.1	56.1	19.0	28.4
	100	86.8	55.0	67.3	68.1	26.7	38.3	56.4	22.1	31.8

Table C.7.: Detailed results on test set for system FL-CONTEXT-PREDS.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
KK-FLAT $k = 1$	10	85.6	41.0	55.4	64.0	11.7	19.8	40.7	7.4	12.6
	25	85.0	44.6	58.5	67.5	15.6	25.3	53.5	12.3	20.0
	50	86.2	48.4	62.0	70.3	19.7	30.8	58.8	16.5	25.8
	100	86.2	54.9	67.1	71.4	25.2	37.3	61.6	21.8	32.2
KK-FLAT $k = 2$	10	84.5	41.2	55.3	65.2	12.4	20.8	42.8	8.1	13.7
	25	84.4	45.2	58.9	65.9	14.6	23.9	52.3	11.6	19.0
	50	84.9	45.7	59.4	69.4	17.5	27.9	57.4	14.4	23.1
	100	85.0	52.5	64.9	70.1	23.3	35.0	60.3	20.1	30.1
KK-FLAT $k = 3$	10	83.1	40.2	54.1	63.6	12.7	21.2	39.5	7.9	13.2
	25	84.6	43.9	57.8	66.4	14.6	23.9	52.6	11.5	18.9
	50	85.4	47.7	61.2	69.0	18.0	28.6	57.4	15.0	23.8
	100	85.3	51.3	64.0	70.3	22.0	33.5	59.3	18.5	28.3
KK-FLAT $k = 4$	10	83.3	42.9	56.6	63.8	12.5	20.9	40.9	8.0	13.4
	25	83.3	42.8	56.5	65.5	14.1	23.2	51.5	11.1	18.2
	50	85.3	45.9	59.7	68.9	17.3	27.6	55.3	13.8	22.1
	100	84.6	48.0	61.2	68.8	20.4	31.5	57.5	17.1	26.3
KK-FLAT $k = 5$	10	82.3	43.8	57.1	64.0	13.2	21.8	41.1	8.4	14.0
	25	83.4	43.2	57.0	65.5	15.0	24.3	51.0	11.6	18.9
	50	84.7	44.6	58.4	68.4	17.0	27.2	54.5	13.5	21.7
	100	82.9	45.4	58.6	69.7	19.9	30.9	57.8	16.5	25.6
KK-FLAT $k = 6$	10	82.3	43.9	57.2	65.6	12.1	20.4	42.2	7.8	13.1
	25	82.7	42.4	56.1	63.7	14.4	23.5	48.9	11.1	18.0
	50	84.0	43.5	57.3	67.7	16.5	26.5	53.6	13.0	21.0
	100	82.9	45.8	59.0	69.9	20.2	31.3	57.0	16.4	25.5
KK-FLAT $k = 7$	10	82.1	43.9	57.2	63.2	11.7	19.7	41.5	7.7	12.9
	25	82.0	42.2	55.7	63.6	13.6	22.4	48.9	10.4	17.2
	50	83.4	43.4	57.1	68.1	16.4	26.4	54.8	13.2	21.2
	100	82.3	45.5	58.6	68.9	19.6	30.5	57.0	16.2	25.3
KK-FLAT $k = 8$	10	82.1	43.9	57.2	63.0	11.7	19.8	41.6	7.7	13.1
	25	82.4	42.3	55.9	62.8	13.7	22.5	48.8	10.7	17.5
	50	83.3	44.1	57.7	68.0	16.9	27.1	54.8	13.6	21.8
	100	81.9	44.4	57.5	68.7	19.4	30.3	55.8	15.8	24.6
KK-FLAT $k = 9$	10	81.2	41.6	55.0	62.2	12.0	20.2	40.3	7.8	13.1
	25	82.6	42.0	55.7	63.5	14.4	23.4	49.3	11.2	18.2
	50	82.7	44.5	57.9	68.3	17.4	27.8	55.4	14.1	22.5
	100	82.3	44.5	57.8	67.9	19.7	30.5	54.9	15.9	24.7
KK-FLAT $k = 10$	10	81.6	41.0	54.6	63.8	12.4	20.8	40.8	7.9	13.3
	25	83.1	42.4	56.1	63.9	14.5	23.7	49.9	11.3	18.5
	50	82.4	44.4	57.7	67.3	17.3	27.5	54.8	14.1	22.4
	100	81.5	44.1	57.3	66.0	19.7	30.3	53.8	16.0	24.7

Table C.8.: Detailed results on test set for system KK-FLAT.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
KK-CONTEXT $k = 1$	10	86.2	44.8	58.9	62.8	13.8	22.6	45.7	10.0	16.5
	25	84.5	49.8	62.7	64.4	18.8	29.2	52.9	15.5	23.9
	50	87.0	55.6	67.9	69.9	24.6	36.4	60.1	21.2	31.3
	100	86.7	59.7	70.7	70.9	30.6	42.7	60.8	26.2	36.7
KK-CONTEXT $k = 2$	10	84.9	44.4	58.3	61.6	13.9	22.6	46.0	10.3	16.9
	25	86.8	47.8	61.6	69.9	18.2	28.9	58.5	15.2	24.1
	50	86.0	53.6	66.1	69.8	24.1	35.8	58.9	20.3	30.2
	100	85.8	57.4	68.7	70.7	29.7	41.8	61.0	25.7	36.1
KK-CONTEXT $k = 3$	10	84.6	46.8	60.3	64.3	14.5	23.7	47.2	10.7	17.4
	25	84.3	51.9	64.3	68.4	19.8	30.7	56.7	16.4	25.5
	50	86.3	56.4	68.2	71.9	25.4	37.6	60.9	21.6	31.8
	100	86.0	57.5	69.0	72.8	29.9	42.4	62.4	25.7	36.4
KK-CONTEXT $k = 4$	10	84.4	47.8	61.1	65.0	15.4	24.9	46.9	11.1	18.0
	25	84.6	51.2	63.7	68.7	20.4	31.4	55.6	16.5	25.5
	50	85.8	56.7	68.2	70.2	26.2	38.2	58.1	21.7	31.6
	100	85.9	57.9	69.2	73.1	30.5	43.0	62.6	26.1	36.9
KK-CONTEXT $k = 5$	10	84.1	47.4	60.6	66.2	16.2	26.1	47.1	11.5	18.5
	25	84.3	51.1	63.6	69.0	20.7	31.9	56.5	17.0	26.1
	50	85.8	57.5	68.9	70.7	26.9	39.0	59.2	22.5	32.6
	100	86.2	57.8	69.2	73.2	30.1	42.7	62.9	25.9	36.6
KK-CONTEXT $k = 6$	10	83.7	47.3	60.4	63.8	16.8	26.7	45.5	12.0	19.0
	25	84.0	50.4	63.0	67.2	20.6	31.6	54.3	16.7	25.5
	50	85.8	58.7	69.7	70.8	27.6	39.7	58.9	23.0	33.1
	100	85.9	58.5	69.6	72.9	30.8	43.3	61.9	26.1	36.7
KK-CONTEXT $k = 7$	10	84.2	46.7	60.0	64.4	16.5	26.3	46.4	11.9	18.9
	25	84.1	49.0	61.9	67.7	20.4	31.3	55.5	16.7	25.7
	50	85.4	57.2	68.5	71.1	27.7	39.9	59.1	23.0	33.1
	100	85.6	58.4	69.5	72.5	31.0	43.4	61.8	26.4	37.0
KK-CONTEXT $k = 8$	10	83.8	46.5	59.8	63.5	16.8	26.5	46.4	12.2	19.4
	25	84.5	49.6	62.5	67.8	20.2	31.1	55.0	16.4	25.3
	50	85.6	56.7	68.2	71.1	27.8	40.0	59.1	23.1	33.2
	100	85.5	58.0	69.1	72.5	31.0	43.4	62.2	26.6	37.2
KK-CONTEXT $k = 9$	10	84.0	46.4	59.8	64.1	16.7	26.5	47.1	12.3	19.5
	25	85.0	51.0	63.8	69.2	21.5	32.8	56.7	17.6	26.9
	50	85.6	55.9	67.7	71.1	27.5	39.6	59.0	22.8	32.9
	100	85.4	56.9	68.3	72.5	30.3	42.7	61.7	25.8	36.4
KK-CONTEXT $k = 10$	10	84.0	45.9	59.3	64.6	17.4	27.5	47.0	12.7	20.0
	25	85.3	49.7	62.8	68.8	20.8	32.0	56.0	16.9	26.0
	50	85.6	56.2	67.9	70.9	27.3	39.5	59.3	22.9	33.0
	100	85.5	56.5	68.0	72.9	30.6	43.1	61.7	25.9	36.5

Table C.9.: Detailed results on test set for system KK-CONTEXT.

System, k	d	Predicate ident.			Argument ident.			Argument class.		
		P	R	F	P	R	F	P_μ	R_μ	F_μ
BASELINE	10	83.4	41.6	55.5	55.0	13.5	21.7	39.9	9.8	15.7
	25	84.1	51.5	63.9	62.1	20.0	30.2	50.6	16.3	24.6
	50	85.4	59.0	69.8	67.6	26.6	38.1	58.2	22.9	32.8
	100	85.0	63.2	72.5	68.2	31.7	43.3	60.0	27.9	38.1
KK-CONTEXT-PREDS $k = 1$	10	83.9	37.4	51.7	57.9	12.0	19.9	40.5	8.4	13.9
	25	85.6	45.5	59.4	66.5	17.9	28.2	55.5	14.9	23.5
	50	86.2	54.2	66.6	69.4	24.9	36.6	58.8	21.1	31.1
	100	87.0	60.1	71.1	71.6	31.0	43.2	61.3	26.5	37.0
KK-CONTEXT-PREDS $k = 2$	10	85.3	39.8	54.3	60.6	13.2	21.7	43.6	9.5	15.6
	25	86.3	45.4	59.5	69.7	18.9	29.8	57.4	15.6	24.5
	50	87.3	53.6	66.4	70.2	25.5	37.4	59.4	21.6	31.7
	100	87.3	59.1	70.5	72.2	31.0	43.3	62.2	26.7	37.4
KK-CONTEXT-PREDS $k = 3$	10	86.4	43.2	57.6	65.9	15.7	25.4	47.3	11.3	18.2
	25	86.9	43.9	58.4	69.4	18.7	29.4	55.8	15.0	23.7
	50	87.4	52.5	65.6	69.7	24.9	36.7	59.2	21.1	31.1
	100	87.3	57.1	69.0	72.6	30.4	42.9	62.1	26.0	36.7
KK-CONTEXT-PREDS $k = 4$	10	85.5	44.2	58.3	66.6	15.9	25.7	45.6	10.9	17.6
	25	86.6	45.5	59.7	70.2	19.4	30.4	56.6	15.7	24.5
	50	87.0	53.3	66.1	70.1	25.3	37.2	58.9	21.2	31.2
	100	86.4	55.9	67.9	72.8	29.6	42.1	61.0	24.8	35.3
KK-CONTEXT-PREDS $k = 5$	10	85.3	42.4	56.6	68.9	15.4	25.1	47.9	10.7	17.5
	25	86.2	43.3	57.6	70.6	18.2	28.9	57.1	14.7	23.4
	50	87.1	51.9	65.1	70.2	24.7	36.6	59.2	20.8	30.8
	100	86.5	55.2	67.4	72.9	28.9	41.4	61.9	24.6	35.2
KK-CONTEXT-PREDS $k = 6$	10	84.1	38.0	52.4	67.2	14.5	23.9	45.7	9.9	16.3
	25	86.7	41.0	55.7	70.6	17.4	27.9	57.4	14.1	22.7
	50	87.4	49.4	63.1	70.5	23.4	35.1	58.7	19.5	29.2
	100	86.1	55.4	67.4	71.8	28.9	41.2	60.1	24.2	34.5
KK-CONTEXT-PREDS $k = 7$	10	84.3	39.9	54.2	66.8	15.8	25.5	47.6	11.3	18.2
	25	87.0	41.9	56.6	70.3	18.1	28.8	57.3	14.7	23.5
	50	87.5	52.2	65.4	70.4	25.1	37.0	58.8	21.0	30.9
	100	85.5	56.0	67.7	71.7	29.4	41.7	59.9	24.5	34.8
KK-CONTEXT-PREDS $k = 8$	10	85.5	41.1	55.5	66.9	16.3	26.2	45.9	11.2	18.0
	25	87.9	41.8	56.6	69.9	18.0	28.6	56.8	14.6	23.3
	50	87.5	52.6	65.7	70.4	25.7	37.6	59.0	21.5	31.5
	100	86.5	55.5	67.6	71.7	29.4	41.7	59.6	24.5	34.7
KK-CONTEXT-PREDS $k = 9$	10	86.3	41.2	55.8	69.0	16.4	26.5	47.0	11.2	18.0
	25	87.8	42.9	57.6	69.8	18.6	29.4	56.6	15.1	23.8
	50	87.1	52.6	65.6	70.2	25.7	37.6	59.0	21.6	31.6
	100	86.3	55.6	67.7	71.4	29.9	42.1	59.3	24.8	35.0
KK-CONTEXT-PREDS $k = 10$	10	86.5	42.8	57.2	67.6	17.2	27.5	45.7	11.7	18.6
	25	87.5	43.3	58.0	70.3	19.1	30.1	56.9	15.5	24.3
	50	86.8	53.3	66.0	70.1	26.1	38.0	58.6	21.8	31.8
	100	85.7	55.7	67.5	70.9	30.0	42.1	59.2	25.0	35.2

Table C.10.: Detailed results on test set for system KK-CONTEXT-PREDS.

Bibliography

- Abend, O., Reichart, R., and Rappoport, A. (2009). Unsupervised argument identification for semantic role labeling. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP (ACL '09)*, pages 28–36.
- Abney, S. (2007). *Semisupervised Learning for Computational Linguistics*. Chapman & Hall/CRC Press.
- Andreevskaia, A. and Bergler, S. (2006). Mining WordNet for fuzzy sentiment: Sentiment tag extraction from WordNet glosses. In *Proceedings of the 11th Conference of the European Chapter of the Association for the Computational Linguistics (EACL '06)*, pages 209–216.
- Baccianella, S., Esuli, A., and Sebastiani, F. (2010). SentiWordNet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In *Proceedings of the 7th conference on International Language Resources and Evaluation (LREC '10)*, pages 2200–2204.
- Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics (ACL '98)*, pages 86–90.
- Bakhshandeh, O. and Allen, J. (2015). Semantic framework for comparison structures in natural language. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP '15)*, pages 993–1002.
- Benamara, F., Chardon, B., Mathieu, Y., Popescu, V., and Asher, N. (2012). How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics (ExProM '12)*, pages 10–18.
- Björkelund, A., Hafdell, L., and Nugues, P. (2009). Multilingual semantic role labeling. In *Proceedings of the 13th Conference on Computational Natural Language Learning: Shared Task (CoNLL '09)*, pages 43–48.
- Blanco, E. and Moldovan, D. (2011). Some issues on detecting negation from text. In *Proceedings of the 24th Florida Artificial Intelligence Research Society Conference (FLAIRS '11)*, pages 228–233.
- Blutner, R. (2004). Pragmatics and the lexicon. In Horn, L. R. and Ward, G., editors, *The Handbook of Pragmatics*, pages 488–514. Blackwell Publishing.
- Bohnet, B. (2010). Very high accuracy and fast dependency parsing is not a contradic-

- tion. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pages 89–97.
- Branavan, S. R. K., Chen, H., Eisenstein, J., and Barzilay, R. (2009). Learning document-level semantic properties from free-text annotations. *Journal of Artificial Intelligence Research*, 34:569–603.
- Brown, P. F., deSouza, P. V., Mercer, R. L., Della Pietra, V. J., and Lai, J. C. (1992). Class-based n -gram models of natural language. *Computational Linguistics*, 18(4):467–479.
- Choi, Y. and Cardie, C. (2008). Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 793–801.
- Cohen, J. (1960). A coefficient of agreement for nominal scales. *Educational and Psychological Measurement*, 20(1):37–46.
- Councill, I. G., McDonald, R., and Velikovich, L. (2010). What’s great and what’s not: Learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP '10)*, pages 51–59.
- Cuzzolin, P. and Lehmann, C. (2004). Comparison and gradation. In v. Booij, G. E., Lehmann, C., and Mugdan, Joachim and Skopeteas, S., editors, *Morphology. An International Handbook on Inflection and Word-Formation*, pages 1212–1220. De Gruyter Mouton.
- Das, S. R. and Chen, M. Y. (2001). Yahoo! for Amazon: Sentiment extraction from small talk on the web. In *Proceedings of the 8th Asia Pacific Finance Association Annual Conference (APFA '01)*.
- Dave, K., Lawrence, S., and Pennock, D. M. (2003). Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. In *Proceedings of the 12th International Conference on World Wide Web (WWW '03)*, pages 519–528.
- Ding, X., Liu, B., and Yu, P. S. (2008). A holistic lexicon-based approach to opinion mining. In *Proceedings of the 2008 International Conference on Web Search and Data Mining (WSDM '08)*, pages 231–240.
- Dragut, E., Wang, H., Yu, C., Sistla, P., and Meng, W. (2012). Polarity consistency checking for sentiment dictionaries. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pages 997–1005.
- Dunning, T. (1993). Accurate methods for the statistics of surprise and coincidence. *Computational Linguistics*, 19(1):61–74.
- Ehrmann, M., Turchi, M., and Steinberger, R. (2011). Building a multilingual named entity-annotated corpus using annotation projection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP '11)*, pages 118–124.

- Fan, R.-E., Chang, K.-W., Hsieh, C.-J., Wang, X.-R., and Lin, C.-J. (2008). LIBLINEAR: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.
- Farkas, R. (2011). Learning local content shift detectors from document-level information. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 759–770.
- Farkas, R., Vincze, V., Móra, G., Csirik, J., and Szarvas, G. (2010). The CoNLL-2010 shared task: Learning to detect hedges and their scope in natural language text. In *Proceedings of the 14th Conference on Computational Natural Language Learning: Shared Task (CoNLL '10)*, pages 1–12.
- Feldman, R., Fresco, M., Goldenberg, J., Netzer, O., and Ungar, L. (2007). Extracting product comparisons from discussion boards. In *Proceedings of the 7th IEEE International Conference on Data Mining (ICDM '07)*, pages 469–474.
- Fillmore, C. (1968). The case for case. In Bach, E. and Harms, R. T., editors, *Universals in Linguistic Theory*, pages 1–88. Holt, Rinehart and Wilson.
- Franco-Penya, H.-H. and Emms, M. (2012). Projecting semantic roles via Tai mappings. In *Proceedings of 11. Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS '12)*, pages 61–69.
- Fürstenau, H. and Lapata, M. (2009). Semi-supervised semantic role labeling. In *Proceedings of 12th Conference of the European Chapter of the Association for Computational Linguistics (EACL '09)*, pages 220–228.
- Fürstenau, H. and Lapata, M. (2012). Semi-supervised semantic role labeling via structural alignment. *Computational Linguistics*, 38(1):135–171.
- Gamon, M. (2004). Sentiment classification on customer feedback data: Noisy data, large feature vectors, and the role of linguistic analysis. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pages 841–847.
- Ganapathibhotla, M. and Liu, B. (2008). Mining opinions in comparative sentences. In *Proceedings of the 22nd International Conference on Computational Linguistics (COLING '08)*, pages 241–248.
- Ganesan, K. and Zhai, C. (2012). Opinion-based entity ranking. *Information Retrieval*, 15(2):116–150.
- Gildea, D. and Jurafsky, D. (2002). Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288.
- Glaser, A. and Schütze, H. (2012). Automatic generation of short informative sentiment summaries. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL '12)*, pages 276–285.
- Glavaš, G. and Šnajder, J. (2015). Construction and evaluation of event graphs. *Natural Language Engineering*, 21(4):607–652.

- Goldberg, A. B., Fillmore, N., Andrzejewski, D., Xu, Z., Gibson, B., and Zhu, X. (2009). May all your wishes come true: A study of wishes and how to recognize them. In *Proceedings of the 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (HLT-NAACL '09)*, pages 263–271.
- Hatzivassiloglou, V. and McKeown, K. (1997). Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and 8th Conference of the European Chapter of the Association for Computational Linguistics (ACL-EACL '98)*, pages 174–181.
- Heine, B. (1997). *Cognitive Foundations of Grammar*, chapter 6, pages 109–130. Oxford University Press.
- Hogenboom, A., van Iterson, P., Heerschop, B., Frasinca, F., and Kaymak, U. (2011). Determining negation scope and strength in sentiment analysis. In *Proceedings of the 2011 IEEE International Conference on Systems, Man, and Cybernetics (SMC '11)*, pages 2589–2594.
- Hou, F. and Li, G.-H. (2008). Mining Chinese comparative sentences by semantic role labeling. In *Proceedings of the 2008 International Conference on Machine Learning and Cybernetics (ICMLC '08)*, pages 2563–2568.
- Hu, M. and Liu, B. (2004). Mining and summarizing customer reviews. In *Proceedings of the 10th ACM SIGKDD international conference on Knowledge Discovery and Data Mining (KDD '04)*, pages 168–177.
- Huang, X., Wan, X., Yang, J., and Xiao, J. (2008). Learning to identify comparative sentences in Chinese text. In Ho, T.-B. and Zhou, Z.-H., editors, *Trends in Artificial Intelligence, Proceedings of the 10th Pacific Rim International Conference on Artificial Intelligence (PRICAI '08)*, volume 5351 of *Lecture Notes in Computer Science*, pages 187–198. Springer Berlin Heidelberg.
- Huddleston, R. (2002). Comparative constructions. In Huddleston, R. and Pullum, G. K., editors, *The Cambridge Grammar of the English Language*, chapter 13, pages 1097–1170. Cambridge University Press.
- Ikeda, D., Takamura, H., Ratinov, L.-A., and Okumura, M. (2008). Learning to shift the polarity of words for sentiment classification. In *Proceedings of the 3rd International Joint Conference on Natural Language Processing (IJCNLP '08)*, pages 50–57.
- Jia, L., Yu, C., and Meng, W. (2009). The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*, pages 1827–1830.
- Jindal, N. and Liu, B. (2006a). Identifying comparative sentences in text documents. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '06)*, pages 244–251.
- Jindal, N. and Liu, B. (2006b). Mining comparative sentences and relations. In *Proceedings of the 21st national conference on Artificial intelligence (AAAI '06)*, pages

- 1331–1336.
- Jindal, N. and Liu, B. (2008). Opinion spam and analysis. In *Proceedings of 2008 International Conference on Web Search and Data Mining (WSDM '08)*, pages 219–230.
- Johansson, R. and Moschitti, A. (2013). Relational features in fine-grained opinion analysis. *Computational Linguistics*, 39(3):473–509.
- Joshi, M. and Penstein-Rosé, C. (2009). Generalizing dependency features for opinion mining. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP '09)*, pages 313–316.
- Kanayama, H., Nasukawa, T., and Watanabe, H. (2004). Deeper sentiment analysis using machine translation technology. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pages 494–500.
- Kennedy, A. and Inkpen, D. (2006). Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125.
- Kennedy, C. (2010). Comparatives, semantics of. In Barber, A. and Stainton, R. J., editors, *Concise Encyclopedia of Philosophy of Language and Linguistics*, pages 68–70. Elsevier.
- Kessler, J. S., Eckert, M., Clark, L., and Nicolov, N. (2010). The 2010 ICWSM JDPA sentiment corpus for the automotive domain. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media Data Workshop Challenge (ICWSM-DWC '10)*.
- Kessler, J. S. and Nicolov, N. (2009). Targeting sentiment expressions through supervised ranking of linguistic configurations. In *Proceedings of the 3rd International AAAI Conference on Weblogs and Social Media (ICWSM '09)*, pages 90–97.
- Kessler, W. (2014). Improving the detection of comparison arguments in product reviews. In Plödereder, E., Grunske, L., Schneider, E., and Ull, D., editors, *Lecture Notes in Informatics*, volume 232, pages 2311–2316. Gesellschaft für Informatik, Bonn.
- Kessler, W., Klinger, R., and Kuhn, J. (2015). Towards opinion mining from reviews for the prediction of product rankings. In *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis (WASSA '15)*, pages 51–57.
- Kessler, W. and Kuhn, J. (2013). Detection of product comparisons – How far does an out-of-the-box semantic role labeling system take you? In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1892–1897.
- Kessler, W. and Kuhn, J. (2014a). A corpus of comparisons in product reviews. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC '14)*, pages 2242–2248.

- Kessler, W. and Kuhn, J. (2014b). Detecting comparative sentiment expressions – A case study in annotation design decisions. In *Proceedings of 12. Konferenz zur Verarbeitung Natürlicher Sprache (KONVENS '14)*, pages 165–170.
- Kessler, W. and Kuhn, J. (2015). Structural alignment for comparison detection. In *Proceedings of the 10th Conference on Recent Advances in Natural Language Processing (RANLP '15)*, pages 275–281.
- Kessler, W. and Schütze, H. (2012). Classification of inconsistent sentiment words using syntactic constructions. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING '12)*, pages 569–578.
- Kim, S.-M. and Hovy, E. (2006). Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of the Workshop on Sentiment and Subjectivity in Text (SST '06)*, pages 1–8.
- Klinger, R. and Cimiano, P. (2013). Bi-directional inter-dependencies of subjective expressions and targets and their value for a joint model. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 848–854.
- Koopmans, T. C. and Beckmann, M. (1957). Assignment problems and the location of economic activities. *Econometrica*, 25(1):53–76.
- Kuhn, H. W. (1955). The Hungarian method for the assignment problem. *Naval Research Logistics*, 2(1-2):83–97.
- Kurashima, T., Bessho, K., Toda, H., Uchiyama, T., and Kataoka, R. (2008). Ranking entities using comparative relations. In Bhowmick, S. S., Küng, J., and Wagner, R., editors, *Database and Expert Systems Applications*, volume 5181, pages 124–133. Springer Berlin Heidelberg.
- Landis, J. R. and Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159–174.
- Li, S., Lee, S. Y. M., Chen, Y., Huang, C.-R., and Zhou, G. (2010). Sentiment classification and polarity shifting. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING '10)*, pages 635–643.
- Li, S., Zha, Z.-J., Ming, Z., Wang, M., Chua, T.-S., Guo, J., and Xu, W. (2011). Product comparison using comparative relations. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '11)*, pages 1151–1152.
- Liu, B. (2015). *Sentiment Analysis*. Cambridge University Press.
- Liu, J. and Seneff, S. (2009). Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, pages 161–169.
- Lluís, X., Carreras, X., and Màrquez, L. (2013). Joint arc-factored parsing of syntactic and semantic dependencies. *Transactions of the Association for Computational*

- Linguistics*, 1:219–230.
- Malmaud, J., Wagner, E., Chang, N., and Murphy, K. (2014). Cooking with semantics. In *Proceedings of the ACL 2014 Workshop on Semantic Parsing (SP '14)*, pages 33–38.
- Manning, C. and Klein, D. (2003). Optimization, maxent models, and conditional estimation without magic. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: Tutorials (HLT-NAACL '03)*.
- Manning, C. D., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S. J., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pages 55–60.
- Moilanen, K. and Pulman, S. (2007). Sentiment composition. In *Proceedings of the 2007 Conference on Recent Advances in Natural Language Processing (RANLP '07)*, pages 378–382.
- Morante, R. and Blanco, E. (2012). *SEM 2012 Shared Task: Resolving the scope and focus of negation. In *Proceedings of the First Joint Conference on Lexical and Computational Semantics: Shared Task (*Sem '12)*, pages 265–274.
- Morante, R., Liekens, A., and Daelemans, W. (2008). Learning the scope of negation in biomedical texts. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing (EMNLP '08)*, pages 715–724.
- Nakagawa, T., Inui, K., and Kurohashi, S. (2010). Dependency tree-based sentiment classification using CRFs with hidden variables. In *Proceedings of the 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics (HLT '10)*, pages 786–794.
- Narayanan, R., Liu, B., and Choudhary, A. (2009). Sentiment analysis of conditional sentences. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing (EMNLP '09)*, pages 180–189.
- Ng, V., Dasgupta, S., and Arifin, S. M. N. (2006). Examining the role of linguistic knowledge sources in the automatic identification and classification of reviews. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL '06)*, pages 611–618.
- Noreen, E. W. (1989). *Computer-intensive methods for testing hypotheses: An introduction*. Wiley & Sons.
- Padó, S. and Lapata, M. (2009). Cross-lingual annotation projection for semantic roles. *Journal of Artificial Intelligence Research*, 36:307–340.
- Page, L., Brin, S., Motwani, R., and Winograd, T. (1999). The PageRank citation ranking: Bringing order to the web. Technical report, Stanford University.
- Palmer, M., Gildea, D., and Kingsbury, P. (2005). The proposition bank: An annotated

- corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.
- Pang, B. and Lee, L. (2005). Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics (ACL '05)*, pages 115–124.
- Pang, B. and Lee, L. (2008). Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.
- Pang, B., Lee, L., and Vaithyanathan, S. (2002). Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP '02)*, pages 79–86.
- Petrov, S., Das, D., and McDonald, R. (2012). A universal part-of-speech tagset. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC '12)*, pages 2089–2096.
- Polanyi, L. and Zaenen, A. (2004). Contextual valence shifters. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text*, pages 106–111.
- Pröllochs, N., Feuerriegel, S., and Neumann, D. (2015). Enhancing sentiment analysis of financial news by detecting negation scopes. In *Proceedings of the 48th Hawaii International Conference on System Sciences (HICSS '15)*, pages 959–968.
- Pullum, G. K. and Huddleston, R. (2002). Negation. In Huddleston, R. and Pullum, G. K., editors, *The Cambridge Grammar of the English Language*, chapter 9, pages 785–850. Cambridge University Press.
- Ramanand, J., Bhavsar, K., and Pedanekar, N. (2010). Wishful thinking: Finding suggestions and 'buy' wishes from product reviews. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Approaches to Analysis and Generation of Emotion in Text (CAAGET '10)*, pages 54–61.
- Rokach, L., Romano, R., and Maimon, O. (2008). Negation recognition in medical narrative reports. *Information Retrieval*, 11(6):499–538.
- Ruppenhofer, J. and Rehbein, I. (2012). Semantic frames as an anchor representation for sentiment analysis. In *Proceedings of the 3rd Workshop in Computational Approaches to Subjectivity and Sentiment Analysis (WASSA '12)*, pages 104–109.
- Ruppenhofer, J., Somasundaran, S., and Wiebe, J. (2008). Finding the sources and targets of subjective expressions. In *Proceedings of the 6th International Conference on Language Resources and Evaluation (LREC '08)*, pages 2781–2788.
- Sayeed, A. B., Boyd-Graber, J., Rusk, B., and Weinberg, A. (2012). Grammatical structures for word-level sentiment detection. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '12)*, pages 667–676.
- Scheible, C. and Schütze, H. (2013). Sentiment relevance. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 954–963.

- Scheible, S. (2010). The smallest, cheapest, and best: Superlatives in opinion mining. In *Proceedings of the 1st Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA '10)*, pages 52–58.
- Shaikh, M. A. M., Prendinger, H., and Mitsuru, I. (2007). Assessing sentiment of text by semantic dependency and contextual valence analysis. In Paiva, A., Prada, R., and Picard, R., editors, *Affective Computing and Intelligent Interaction, Proceedings of AII*, volume 4738 of *Lecture Notes in Computer Science*, pages 191–202. Springer Berlin Heidelberg.
- Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C., Ng, A., and Potts, C. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing (EMNLP '13)*, pages 1631–1642.
- Spearman, C. (1904). The proof and measurement of association between two things. *American Journal of Psychology*, 15(1):72–101.
- Staab, S. and Hahn, U. (1997). Comparatives in context. In *Proceedings of the 14th National Conference on Artificial Intelligence and 9th Conference on Innovative Applications of Artificial Intelligence (AAAI-IAAI '97)*, pages 616–621.
- Stassen, L. (2013). Comparative constructions. In Dryer, M. S. and Haspelmath, M., editors, *The World Atlas of Language Structures Online (WALS Online)*, chapter 121. Max Planck Institute for Evolutionary Anthropology.
- Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245–251.
- Stone, P. J., Dunphy, D. C., Smith, M. S., and Ogilvie, D. M. (1966). *General Inquirer: A Computer Approach to Content Analysis*. The MIT Press.
- Swier, R. S. and Stevenson, S. (2004). Unsupervised semantic role labelling. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing (EMNLP '04)*, pages 95–102.
- Swier, R. S. and Stevenson, S. (2005). Exploiting a verb lexicon in automatic semantic role labelling. In *Proceedings of the 2005 Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP '05)*, pages 883–890.
- Taboada, M., Brooke, J., Tofiloski, M., Voll, K., and Stede, M. (2011). Lexicon-based methods for sentiment analysis. *Computational Linguistics*, 37(2):267–307.
- Tkachenko, M. and Lauw, H. W. (2014). Generative modeling of entity comparisons in text. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14)*, pages 859–868.
- Tkachenko, M. and Lauw, H. W. (2015). A convolution kernel approach to identifying comparisons in text. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP '15)*, pages 376–386.

- Tsur, O., Davidov, D., and Rappoport, A. (2010). ICWSM – a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *Proceedings of the 4th International AAAI Conference on Weblogs and Social Media (ICWSM '10)*, pages 162–169.
- Tu, Z., He, Y., Foster, J., van Genabith, J., Liu, Q., and Lin, S. (2012). Identifying high-impact sub-structures for convolution kernels in document-level sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pages 338–343.
- Turney, P. D. and Littman, M. L. (2003). Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346.
- Wang, S. and Manning, C. (2012). Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL '12)*, pages 90–94.
- Wiegand, M., Balahur, A., Roth, B., Klakow, D., and Montoyo, A. (2010). A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing (NeSp-NLP '10)*, pages 60–68.
- Wilson, T., Wiebe, J., and Hoffmann, P. (2005). Recognizing contextual polarity in phrase-level sentiment analysis. In *Proceedings of the 2005 Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT-EMNLP '05)*, pages 347–354.
- Wilson, T., Wiebe, J., and Hoffmann, P. (2009). Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, 35(3):399–433.
- Xia, R., Xu, F., Yu, J., Qi, Y., and Cambria, E. (2016). Polarity shift detection, elimination and ensemble: A three-stage model for document-level sentiment analysis. *Information Processing & Management*, 52(1):36–45.
- Xu, K., Liao, S. S., Lau, R. Y. K., Tang, H., and Wang, S. (2009). Building comparative product relation maps by mining consumer opinions. In *Proceedings of the 15th Americas Conference on Information Systems (AMCIS '09)*, pages 1653–1661.
- Xu, K., Liao, S. S., Li, J., and Song, Y. (2011). Mining comparative opinions from customer reviews for competitive intelligence. *Decision Support Systems*, 50(4):743–754.
- Yang, S. and Ko, Y. (2009). Extracting comparative sentences from Korean text documents using comparative lexical patterns and machine learning techniques. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the Association for Computational Linguistics and the 4th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP '09)*, pages 153–156.
- Yang, S. and Ko, Y. (2011a). Extracting comparative entities and predicates from texts

- using comparative type classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT '11)*, pages 1636–1644.
- Yang, S. and Ko, Y. (2011b). Finding relevant features for Korean comparative sentence extraction. *Pattern Recognition Letters*, 32(2):293–296.
- Yao, X., Van Durme, B., Callison-Burch, C., and Clark, P. (2013). A lightweight and high performance monolingual word aligner. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (ACL '13)*, pages 702–707.
- Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics (ACL '95)*, pages 189–196.
- Yarowsky, D., Ngai, G., and Wicentowski, R. (2001). Inducing multilingual text analysis tools via robust projection across aligned corpora. In *Proceedings of the 1st International Conference on Human Language Technology Research (HLT '01)*.
- Zhang, K., Narayanan, R., and Choudhary, A. (2009). Mining online customer reviews for ranking products. Technical report, Center for Ultra-scale Computing and Information Security, NorthwestUniversity.
- Zhang, K., Narayanan, R., and Choudhary, A. (2010). Voice of the customers: Mining online customer reviews for product feature-based ranking. In *Proceedings of the 3rd Workshop on Online Social Networks (WOSN '10)*.
- Zhang, Z., Guo, C., and Goes, P. (2013). Product comparison networks for competitive analysis of online word-of-mouth. *ACM Transactions on Management Information Systems (TMIS)*, 3(4):20:1–20:22.
- Zhu, X., Guo, H., Mohammad, S., and Kiritchenko, S. (2014). An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (ACL '14)*, pages 304–313.
- Zirn, C., Niepert, M., Stuckenschmidt, H., and Strube, M. (2011). Fine-grained sentiment analysis with structural features. In *Proceedings of 5th International Joint Conference on Natural Language Processing (IJCNLP '11)*, pages 336–344.