ABSTRACT

Title of dissertation: A DISTRIBUTIONAL AND SYNTACTIC APPROACH TO FINE-GRAINED OPINION MINING

Asad Basheer Sayeed, Doctor of Philosophy, 2011

Dissertation directed by: Professor Amy Weinberg
Department of Linguistics,
Department of Computer Science,
and Institute for Advanced Computer Studies

This thesis contributes to a larger social science research program of analyzing the diffusion of IT innovations. We show how to automatically discriminate portions of text dealing with opinions about innovations by finding {source, target, opinion} triples in text. In this context, we can discern a list of innovations as targets from the domain itself. We can then use this list as an anchor for finding the other two members of the triple at a "fine-grained" level—paragraph contexts or less.

We first demonstrate a vector space model for finding opinionated contexts in which the innovation targets are mentioned. We can find paragraph-level contexts by searching for an "expresses-an-opinion-about" relation between sources and targets using a supervised model with an SVM that uses features derived from a general-purpose subjectivity lexicon and a corpus indexing tool. We show that our algorithm correctly filters the domain relevant subset of subjectivity terms so that they are more highly valued.

We then turn to identifying the opinion. Typically, opinions in opinion mining are

taken to be positive or negative. We discuss a crowd sourcing technique developed to create the seed data describing human perception of opinion bearing language needed for our supervised learning algorithm. Our user interface successfully limited the meta-subjectivity inherent in the task ("What is an opinion?") while reliably retrieving relevant opinionated words using labour not expert in the domain.

Finally, we developed a new data structure and modeling technique for connecting targets with the correct within-sentence opinionated language. Syntactic relatedness tries (SRTs) contain all paths from a dependency graph of a sentence that connect a target expression to a candidate opinionated word. We use factor graphs to model how far a path through the SRT must be followed in order to connect the right targets to the right words. It turns out that we can correctly label significant portions of these tries with very rudimentary features such as part-of-speech tags and dependency labels with minimal processing. This technique uses the data from the crowdsourcing technique we developed as training data.

We conclude by placing our work in the context of a larger sentiment classification pipeline and by describing a model for learning from the data structures produced by our work. This work contributes to computational linguistics by proposing and verifying new data gathering techniques and applying recent developments in machine learning to inference over grammatical structures for highly subjective purposes. It applies a suffix tree-based data structure to model opinion in a specific domain by imposing a restriction on the order in which the data is stored in the structure.

A DISTRIBUTIONAL AND SYNTACTIC APPROACH TO FINE-GRAINED OPINION MINING

by

Asad Basheer Sayeed

Dissertation submitted to the Faculty of the Graduate School of the
University of Maryland, College Park in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
2011

Advisory Committee:
Professor Amy Weinberg, Chair/Advisor
Professor William Idsardi, Dean's Representative
Assistant Professor Jordan Boyd-Graber
Assistant Professor Hal Daumé III
Professor Donald Perlis

Dedication

In loving memory of my mother's paternal aunt, Ashraf-un-Nisa Shakir (1919-2009), my Nani, a person of genuine piety, wisdom, erudition, and strong opinions.

Acknowledgements

This is quite likely the highest degree that I will ever receive and the peak of my current educational path. There are a large number of people to thank for having helped me reach this point and having been crucial influences on my life and thinking, and this is an appropriate time to thank as many of them as I can, stretching well back into my previous educational history. If there is anyone I have forgotten, my profound apologies.

First and foremost, I would like to thank my parents, Kabeer Sayeed and Nighat Mehkeri, for bringing me up capable of making use of the opportunities that I have been given. Then I would like to thank my brothers, Wajid, MD, and Safi for being all-around hilarious and fun throughout this entire process. Beyond them, I would like to thank my large extended family that ensures I have a place of welcome almost anywhere I go, and also my many elder and late relatives such as my paternal grandparents Justice Basheer Ahmed Sayeed and Fatima Akhtar Sayeed, as well as my maternal grandparents Mohammed Asadullah Mehkari and Saeed-un-Nisa Mehkari.

Then I would like to thank my many friends from my Canadian life. I will start with the MBL gang from my high school, Lisgar Collegiate Institute in Ottawa. Daniel Kekez kept the Space Simulation dream alive through his on-going career. Soon-to-be-Dr. Terri Oda has had a career rather parallel to mine in many ways and was a source of mutual support and friendship. Dr. Jordan Melzer provided invaluable advice and experience during crucial turns in my doctoral journey. And thanks to Kathleen Blakely, Jamie Rosen, Nic Chop, Adam Elliott, Sarah Mark, Cameron Morland, Neil Leathers, Meghan Roberts, Andria Green, Dr. Margaret McCarthy, Marlene and Andrew Mckay,

and Geoffrey Oakham. And a big thank you to the very many people from Space Sim, the various orchestras and bands and choirs in which I used to perform, whose combined input set me on the path I have followed.

I continue to be in contact with people from Lisgar whose friendship has been invaluable, including Eric Quon-Lee who is now himself moving to the USA. Dr. Alexander "Sasha" Wait gave me my first computer science job. I would also like to thank many teachers who helped me on my way. Mr. Lamperd, Mrs. Poetschke, Mr. Pritchett, Mr. Peters, and others helped me become a better scientist and a better writer. Teachers like Mr. Porter and Mrs. Bradley helped me acquire a more balanced personality by teaching me how to perform and appreciate music. On the language front, Mme. Charron, Mr. Taylor, Herr Paryas, and Frau Abbas helped me gain an appreciation of the differences between languages and planted the seed in my mind that germinated and made me a linguist.

Outside of high school, important people in my intellectual life include Dr. Barbara F. McManus, emerita of the College of New Rochelle, and Dr. Suzanne Bonefas, now at Rhodes College, who brought me into the world of internet-enabled language learning, particular of classical languages via their VRoma Project. Dr. Patrick Lam, now at the University of Waterloo, helped guide my feet towards this Ph.D. He and other friends such as Marie-Pascale Desjardins, Chris Jones, and Emily Johnston remain in contact with me and continue to provide friendship and support.

My Master's degree was an integral part of my path to this dissertation. Dr. Stan Szpakowicz was a patient and open-minded Master's advisor, and I remain in regular contact with him in his University of Ottawa office. Other friends from this period include Dr. Viviane Nastase and Dr. Marina Sokolova. I am also friends with current students such

as Alastair Kennedy.

Now I finally turn to my American life, and my move to the University of Maryland's Ph.D. program. Jim Menasian was an excellent landlord and housemate for the first six years of my University of Maryland, College Park (UMCP) career, and though I have moved out of his basement apartment for the final year of my Ph.D., I continue to visit his home and attend his parties regularly. He has introduced me to a large number of new friends, including the cream of the crop of the DC-area Armenian-American community.

Despite being in the computer science Ph.D. program, I very quickly latched on to the social and intellectual life of the UMCP linguistics department. Linguistics graduate student friends I made include Rebecca McKeown, Dr. Pritha Chandra, Dr. Lydia Grebenyova, Dr. Heather Taylor, Dr. Ellen Lau, Dr. Ivan Ortega-Santos, Dr. Usama Soltan, Dr. Nobue Mori, Dr. Chizuru Nakao, Dr. Masaya Yoshida, Dr. Tim Hunter, Dr. Stacey Conroy, Tim Hawes, Dr. Ilhan Cagri, Dr. Lisa Pearl, Michael Subotin, Dr. Johannes Jurka, and many others. Dr. Atakan Ince remains in the DC area, and he and I discuss matters linguistic regularly over tea and at DC museums. Dr. Ilknur Oded and Dr. Shiti Malhotra defended their dissertations shortly before me. Yakov Kronrod and Darryl McAdams continue to keep the faith. Dr. Chris Dyer was my mirror image, a linguistics student closely interested in the nitty-gritty of practical computer science. Dr. Yuval Marton was an excellent officemate and ongoing, very helpful friend.

Computer science and computational linguistics graduate student friends include Martin Paraskevov, Dr. Adam Lopez, Nick Frangiadakis, Ed Kenschaft, Dr. Okan Kolak, Dr. Matthew Snover, Nate Waisbrot, Dr. David Zajic, Vladimir Eidelman, Denis Filimonov, Zhongqiang Huang, Raul Guerra, Eric Hardisty, Craig Murray, Lidan Wang, Hassan

Sayyadi, Jagadeesh Jagarlamudi, Yuening Hu, Amit Goyal, and Jiarong Jiang. Dr. Nitin Madnani and I still discuss linguistics and technology in general over Hollywood movies. I remain long-time friends with Dr. Alexander Grushin and David Mihalcik whom I met early on in the computer science program.

UMCP post-doctoral researchers have also had a large effect on my Ph.D. career. These include Dr. Saif Mohammad, Dr. Smaranda Muresan, Dr. Tamer Elsayed, Dr. Hendra Setiawan, Dr. Kristy Hollingshead, and Dr. Earl Wagner.

While my dissertation fits within applied computational linguistics, I was and am an avid student of linguistic theory, particularly syntax. The environment in UMCP linguistics was ideal for someone with interests in both areas. I would like to thank the entire faculty of the UMCP lingustics department for their time and advice in helping me stay abreast of developments in lingustics. In particular, Dr. Juan Uriagereka had a strong influence on the way I look at the deeper questions of syntax, and I learned very much from Dr. Norbert Hornstein and Dr. Howard Lasnik in these areas as well. Dr. Philip Resnik showed me how to stay grounded in "the computational" while continuing to pursue my interests in these kinds of abstract questions, and his help and advice in how to pursue a corpus-based social science project was crucial to my successful completion of this degree. Dr. Resnik was also a member of my proposal committee.

I was a member of the Laboratory for Computational Linguistics and Information Processing (CLIP) at UMCP, and I partook of the excellent cross-disciplinary resources represented by th faculty there, including my advisor and Dr. Resnik. I would like to thank Dr. Douglas Oard, Dr. Louiqa Raschid, Dr. Bonnie Dorr, Dr. Mary Harper, Dr. Jimmy Lin, and Dr. Judith Klavans for sharing their wisdom and experience with me. Dr. Alan

Sussman sat on my proposal committee.

The computer science (CS) department at UMCP was also where I learned to teach. I worked as a teaching assistant for Dr. Jeff Foster, Dr. Rance Cleveland, Larry Herman, Dr. Jandelyn Plane, and Dr. Jeff Hollingsworth, and gained a great deal of experience thereby. UMCP CS has a large number of excellent undergraduate student who collaborated with me on some of the papers that constitute part of this dissertation as well as other related work. These are Tiffany Chao, Timothy Meyer, Hieu Nguyen, Olivia Buzek, Sam Blitzstein, Martin Petrov, and Brian Rusk. Computer science administrative staff, such as Fatima Bangura, Jenny Story, Kathy Barton, Arlene Schenk and Jodie Gray made life very much easier for me at UMCP.

Much of the funding for this work came from a project at UMCP's College of Information Studies (iSchool). This project was funded by the National Science Foundation under grant IIS-0729459. Faculty member Dr. Ping Wang provided much of the impetus and motivation for the work in this dissertation, and Dr. Ken Fleischmann provided some of his experience and advice. Non-CS students who contributed by way of the PopIT project include Karen Viruez-Munoz and Andrea Farina. Fellow-traveller PopIT graduate students include Dr. Chia-Jung Tsui and An-Shou Cheng.

I gained important and relevant skills during the program from a couple of internships that I held at IBM's T. J. Watson Research Center. In 2007, I worked under the direction of Dr. Young-Suk Lee, Dr. Yaser Al-Onaizan, and Dr. Salim Roukos on the use of grammar in machine translation. In 2008, I worked with Dr. Soumitra "Ronnie" Sarkar, Dr. Rafah Hosn, Nithya Rajamani, Ruchi Mahindru, and Dr. Yu Deng on an application of corpus statistics to a business problem. The skills gained in this work were crucial to the

completion of this dissertation. I also gained similarly relevant skills in 2008 working on the NIST ACE evaluation with a very large number of people from UMCP, Johns Hopkins University, and BBN Technologies such as Dr. Jim Mayfield and Dr. Christine Piatko, as well as Dr. Nikesh Garera, Dr. David Yarowsky, and many others.

Outside of the university, I had considerable support from some DC-area friends. I would like to thank the DC-area Scrabble Meetup for all the fun times and the opportunities to unwind. It was a very necessary "stabilizer" in this entire process. I would like specifically to mention Ruthie Franco, Julia Wolz, Charles "Steve" Padgett, Malcolm Kenton, and Karen Carter, but the Scrabble gang is large and has many very appreciated faces.

Once again, in all of the above, if I have missed someone important, I apologize. I would also like to take the opportunity to acknowledge all of my online friends from the discussion groups of which I am a member. Some of them I have come to know offline and occasionally meet them. They kept me thinking and exercising my mind while exposing me to new perspectives.

Now I turn to the committee. I would like to thank Dr. William Idsardi, Dr. Hal Daumé, and Dr. Donald Perlis for their kind patience, attention, and attendance at my defense. Dr. Jordan Boyd-Graber was particularly involved in this work, and he helped me develop and organize chapter 4.

I am saving the best for last: my advisor, Dr. Amy Weinberg. Dr. Weinberg dutifully read and commented in detail on any number of drafts of any number of related submissions, including papers, proposals and so on. She dealt with the occasional behind-the-scenes emergencies with alacrity and aplomb. She collaborated closely with me on an

ongoing syntax project. She steered me towards the opportunities to acquire skills that I only now realize in hindsight will be necessary to do all the things I want to do in my future career. She put up with my occasional studently bouts of existential angst and self-doubt, she provided emotional support through one or two personal crises that happened during my time in the program, and she brought me intact to the completion of my Ph.D. Thanks, Amy.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 The sentiment triangle: sources, targets, and opinions

The terms "sentiment analysis" and "opinion mining" cover a wide body of research on and development of systems that can automatically infer certain kinds of emotional states from text (after Pang and Lee [2008] we use the two names interchangeably). The texts in question have ranged from online product reviews to news articles and, more recently, social media such as blog posts. Sentiment analysis for texts such as product reviews has largely focused on recovering rating information from whole reviews. In these kinds of texts, the source of the opinion is known as well as the object about which the opinion is held (a.k.a. the target).

In other kinds of texts, the task becomes more complex. Sometimes, there are multiple conflicting opinions issued by multiple entities about multiple things. Newswire text is an obvious case. An article on politics can contain direct and indirect quotes from multiple politicians. In our work, we focus on another domain which shares this characteristic: the IT business press, in which persons and organizations express opinions about technology, particularly as it affects the adoption of new technological innovations.

This form of sentiment analysis research focuses on the automatic discovery of all components in the expression of an opinion. There are three major components in an opinion-expressing relationship. The first is the source of the opinion, the second is the

target of the opinion, and the last is the semantic content of the opinion itself.

For simplicity's sake, most current work expresses the semantic content of the opinion in terms of polarities: positive or negative sentiment, sometimes neutral. While this is a very simple way of capturing human feelings, it allows us to abstract away from the fine-grained classification task entailed by a fuller taxonomy of private states [Wilson and Wiebe, 2005]. Our work offers an improved solution to the more bounded problem of determining the presence of and participants in an opinion expression.

Existing work on detecting opinion components can best be characterized by seeing the participant types as sides of a triangle, with the vertices reflecting the types of technologies that are the subjects of current research. Source detection alone reduces to the problem of named-entity detection if not accompanied by detection of the opinion. It is similarly the case with target detection. Opinion polarity detection reduces to marking words with dispositions from thesauri [Stone and Hunt, 1963] if we do not take into account the context of the opinion.

If we define the full problem of opinion mining to be that of finding sets of triples of the form $\{$source, target, opinion$\}$, and we recognize an interdependence between the constituents of a given triple, then we can break the problem of finding all three into natural subproblems: finding pairs of them. There is existing work in discovering all three pairs: finding sources and sentiments together, finding sentiments and targets, and finding sources and targets.

1.2 Contributions

Particular attention has been paid in the literature to finding sources and associated opinions. Therefore, in this work, we focus on the other vertices of the triangle: opinion-target identification, and sentiment-target relation extraction.

Our work contains two components that individually expand the box of tools for fine grained sentiment analysis. The first component is our work on a system that uses light information extraction techniques to mine IT business press texts for an "expresses-an-opinion-about" relation (chapter 2).

The second component (chapters 3 and 4) consists of our procedure for opinion-target identification and within-sentence discovery of opinion words and their syntactic relationships to targets. Syntactic relationship discovery gives us added precision in distinguishing the main opinion-bearing words related to a target from opinion words that are unrelated to a target. We present these components as separate but related research efforts; rather than tie them immediately together, we investigate their individual characteristics. In chapter 5, we discuss how to tie them together into a single pipeline as we conclude this work.

Our work requires a resource for opinion word-target links, and we build one by deploying a crowdsourcing technique as discussed in chapter 3. We propose an algorithm that uses dependency parsing and machine learning techniques to identify opinionated mentions of targets and the relevant within-sentence opinion words (chapter 4). Both of these things we do in the context of the identification of sentiment in the IT business press. This is an added challange, because opinion detection in this genre necessarily has subtle

semantic and pragmatic differences from the approach taken by other work in this area on general newspaper text.

The source-target relation extraction task provides the basis for the first part of a sentiment-detection pipeline. This permits us to detect opinion expressions at a large scale. Then we use those detected opinions as information for the more fine-grained target detection task.

Ultimately, we present an approach to the expansion of existing techniques in opinion-mining feature engineering. Alm [2011] recently argued:

> ...there are evaluation concepts in computational linguistics that, at least to some degree, detract attention away from how subjective perception and production phenomena actually manifest themselves in natural language. In encouraging a focus on efforts to achieve "high-performing" systems (as measured along traditional lines), there is a risk involved—the sacrificing of opportunities for fundamental insights that may lead to a more thorough understanding of language uses and users.

The previous work we discuss in section 1.3 mostly focuses directly on improving opinion retrieval and classification scores based on particular and established ways of conceptualizing opinion. We seek to respond to Alm's challenge by stepping back and instead focusing on the necessary groundwork for developing an underlying and very fine-grained conceptualization of the feature space that provides the evidence base for opinionated phenomena in language. Our contributions use efforts to improve retrieval performance in order to extract meaningful features rather than focusing directly on improving retrieval

performance at all costs.

These contributions will improve over existing work by exploiting natural divisions in the problem of detecting opinion components at a fine-grained level over many documents. Our technique of learning opinion-target relations from dependency graphs is intended to enable the discovery of bounds of opinion expressions in a way that preserves syntactic coherence and semantic compositionality. Existing work [Moilanen and Pulman, 2007] has shown that finding the bounds of opinion expressions will improve the accuracy of opinion polarity classification.

However, as these techniques are processor-intensive, our lightweight source-target detection technique allows us to find the general vicinities of opinion expressions, enabling us to avoid parsing every sentence in an entire corpus during opinion polarity classification.

We also generate new resources for training opinion mining systems. We make use of the information technology business press and develop both in-house and crowdsourced annotation techniques in order to produce training data for the supervised learning methods we incorporate into this work.

1.2.1 Fine-grained sentiment analysis

Since the context of our work is improving the sentence-level detection of sentiment, we need to ground the idea of sentiment and opinion in the entity mentions and opinion words that may occur. For example, consider the following sentence from a major information technology (IT) business journal:

Lloyd Hession, chief security officer at BT Radianz in New York, said that

virtualization also opens up a slew of potential network access control issues.

There are three entities in the sentence that have the capacity to express an opinion: Lloyd Hession, BT Radianz, and New York. These are potential opinion *sources*. There are also a number of mentioned concepts that could serve as the topic of an opinion in the sentence, or *target*. These include all the sources, but also "virtualization", "network access control", "network", and so on.

Our task, therefore, is to discriminate between these mentions and choose the ones that are relevant to the interests of the user. Furthermore, such a system must also give some indication of the content of the *opinion* itself. This means that we are actually searching for all triples $\{$*source, target, opinion*$\}$ in this sentence [Kim and Hovy, 2006], and, by extension, the document and the corpus.

The sentence above tells us that Lloyd Hession expressed a view of virtualization, an information technology, that contains a negative opinion, so we have one candidate triple: $\{$Lloyd Hession, virtualization, negative$\}$. Our task is to determine the evidence that an automated system could use to infer that this candidate triple is a member of the class.

There is considerable work on identifying the source of an opinion. In the above sentence, the evidence that Lloyd Hession is a source is the use of "said". It is possible to perform source identification through such means—machine learning over grammatical and lexical cues—with reasonably high accuracy [Choi et al., 2005]. We discuss the existing literature in source identification in section 1.3.2.

However, it is much harder to find obvious features that tell us whether "virtualization" is the target of an opinion. The most recent target identification techniques use machine

learning to determine the presence of a target from known opinionated language [Jakob and Gurevych, 2010]. But in certain restricted domains, it is possible to know what types of targets may be of interest to the user, either because the domain is small enough that expert knowledge can provide the list or because the user is explicitly providing the target through a query.

Even with this information, we still must address the problem of deciding whether or not there was an opinion expressed whenever a target is mentioned, since not all target mentions will necessarily be accompanied by opinion expressions. In the above sentence, we could say that the negative opinion about virtualization is expressed by the words "slew" and "issues." The presence of target foreknowledge reduces the problem to choosing the applicable words from the sentence.

Finally, knowledge that "slew" and "issues" tend to be negative words can assist us with identifying that a negative opinion is being expressed about virtualization, but what if the sentence ended like this: "...a slew *of falsehoods about* potential network access control issues?" We need to build a model that connects "falsehoods" to "slew" and "issues."

To address problems like these, we must draw on the grammatical relationships between pairs. While these relationships can often be revealed through existing parsing technologies, they are usually not direct, as there are often intervening items. Given the variability of language, it is impossible to assemble a complete catalog of the intervening items, but examples include negation and intensifiers, which can change the expressed opinion, and sentiment-neutral words which fill syntactic or stylistic needs. We cope with the variability of expression using machine learning to generalize across observations and

learn which features best enable us to identify opinionated language.

1.2.2 Pragmatic opinion and metasubjectivity

In this section, we offer a pragmatic defintition of opinion mining that covers the domain (in the mathematical sense) of acceptable text spans we can label as bearing opinions, the range of possible labels, and a pragmatic definition of what opinion actually is. While somewhat theoretical, this is important for our annotation efforts in Section 4.4.2, which depend on objective assesments of which statements are opinion bearing.

What can express an opinion? For this work, we define opinion mining (sometimes known as sentiment analysis) above to be the retrieval of a triple {*source, target, opinion*}. We call opinion mining "fine-grained" when we are attempting to retrieve potentially many different {*source, target, opinion*} triples per document. This is particularly challenging when there are multiple triples even at a sentence level. It is further contingent on what we regard to be an opinion, and this is itself contingent on our perspective. In chapter 4, we show that this set of contingencies suggests concentrating on grammatical relations between targets and opinion words rather than also trying to establish relationships between sources and opinion words in a single step..

What is the range of an opinion? In much of the recent literature on automatic opinion mining, *opinion* is at best characterized as a gradient between positive and negative or a binary classification thereof; further complexity affects the reliability of machine-learning techniques [Koppel and Schler, 2006]. Nevertheless, even this kind of binary classification

is fraught with complications.

What is an opinion anyway? We are attempting to retrieve word-level opinion-relevant features, but what words and features are relevant differ across genres and applications. Recent efforts in opinion mining [Ruppenhofer et al., 2008] technology have often tended to take the position that opinion is an internal characteristic of the speaker: a "private state." But this may not always be appropriate to all circumstances. We refer to variation in the application-specific interpretation of the concept of opinion as "metasubjectivity."

In applications of opinion mining that attempt to track the relationshp of opinions in a corpus to external trends such as stock market prediction [Bollen et al., 2010], the actual opinionated state of the source of the opinion not only matters less, but is hard to gauge. The virtualization example contains Lloyd Hession as a potential opinion source, and his *prima facie* disposition is negative. But it could just as well be the case that Mr. Hession prefers that there be problems with virtualization so that he could sell an alternative product or paradigm. This requires considerable world knowledge to disentangle.

Furthermore, there are particular actions or statements [Austin, 1962] that have an effect on the real world, and could be said by implication to express an opinion. Buying and selling in certain contexts implies confidence in a product or service. We can refer to this as "pragmatic opinion" [Somasundaran and Wiebe, 2009].

Since the goal of our exercise is to ascertain the correlation between the source's behaviour and that of others, it may be more appropriate to look at opinion analysis with the view that what we are attempting to discover are the views of an aggregate reader who may otherwise have an interest in the IT concept in question. This reorients our work in

sentiment in a manner that is more useful for applications such as market prediction.

The problem of metasubjectivity and the variability of pragmatic contexts calls for techniques that make the fewest possible advance commitments about the kinds of expressions we are trying to extract. We also need to be able to extract and exploit the maximum variety of features available to enable the broadest range of techniques. We approach this challenge by mining at a *very* fine-grained level: that of words and grammatical relationships.

1.3 Existing work

1.3.1 Context

As we discussed above, we have subdivided the problem of finding {source, target, opinion} triples into finding pairs of the three. In reality, we only need to find two pairs out of three, to have all the elements. Each pair allows us to solve a simpler problem than finding all three at once. The pairs we have selected are the source-target pair and the target-opinion pair.

Most current work only loosely recognizes this division. The preponderance of work focuses on finding opinion sources. At first glance, this makes sense: sources often have obvious syntactic relationships with targets ("John really hates..."). However, Ruppenhofer et al. [2008] point out that this limitation on syntactic contexts establishes an upper bound on source discovery in that there are still categories of sources in very complex syntactic relations, which sometimes persist extra-sententially.

The work we cite in the following sections represents a sample of recent efforts in

fine-grained sentiment analysis, which is a relatively new field. As such, while most of this work is occupied with finding sentiment components such as sources and targets, it generally tends to define the task in ways that are not directly comparable to the work we describe in later chapters. For example, most of it relies on a form of human-annotated training data that bounds opinion expressions in a much more liberal manner than we require in our techniques. We cite this work to give the reader a sense of what technologies are currently in use; in later chapters, we discuss the limitations of these technologies.

1.3.2 Extracting opinions with their sources

Work on opinion source mining has assumed that resource-heavy techniques—heavy in feature engineering, machine learning, or pattern matching technology—suffice not only to find sources and opinions, but also to find targets. We contend in chapter 2 that using a lightweight technique captures more than what can be caught using these heavier techniques. What follows, therefore, is the discussion of recent developments in heavy techniques, with some mention of its shortcomings.

For the past decade, there has been a well-populated program of research focused on detecting opinions along with their sources at the sentence level. Opinion sources tend to be external to the opinion itself, sometimes standing in a syntactic relation to the opinion as, for instance, the subject of a verb of expression ("**The President** believes that.....). This leads to the deployment of particular types of pattern extraction to accomplish this task, usually in the context of some form of clustering or classification.

A preliminary effort is Bethard et al. (2004). They use semantic role labelling,

syntactic information, and lexical information as input to an SVM classifier that detects sentential complements that contain opinion propositions ("I think that **the fish tastes bad**"). Their performance is relatively poor, however, but they established an approach refined by later work.

One of the problems in opinion source identification, particularly from an opinion summarization and question-answering perspective, is that of coreference; for example, pronominal references must be resolved for an opinion summary to make any sense. Stoyanov and Cardie (2006) use a clustering technique based on structured rule learning to construct coreference chains using information derived particularly from the contexts in which opinion sources appear. Our work described in 2 evades some of the need for coreference detection by looking at paragraph contexts, so that it is more likely to find the actual entity mention participating in an opinion expression without resolving pronouns.

Kim and Hovy (2005) detect opinion sources for a question-answering application by ranking candidate pairs of sources and opinion expressions via syntactic features in an maximum entropy model. They can obtain very high accuracy (up to 90%) if they are allowed to consider the three top-ranked results f/rom their system.

One of the major sources of data for sentiment analysis research is the Multi-Perspective Question-Answering (MPQA) corpus; the original annotation effort is described in Wilson and Wiebe (2003)[1].

Choi et al. (2005) experiment with extraction pattern recognition and conditional random field (CRF) based methods to recover opinion source information when the

[1] Our techniques in the rest of the thesis are supervised as most sentiment analysis research has been to date, so we will be discussing annotation efforts throughout this document.

opinions themselves have already been marked in the MPQA corpus. Choi et al. (2006) use a CRF-based method alongside Integer Linear Programming (ILP) and semantic role labelling to extract sources and opinions at the same time.

The original version of the MPQA contained information about opinion sources, but not opinion targets; this was rectified in a much more recent version in 2008. MPQA is the training data used in one system known as OpinionFinder [Wilson et al., 2005a], which performs subjectivity classification, source identification, and polarity classification as a highly integrated process. The technique of Choi et al. (2006) is used for source identification in OpinionFinder.

Both the Kim and Hovy work and the Choi et al. research efforts use highly constraining grammatical knowledge for source detection. We intend to show that we can acquire sources without these efforts—and, by applying heavier techniques to target detection, still detect the localities of opinions.

1.3.3 Extracting opinions with their targets

Finding the targets of opinions is less well-trodden ground at the sentence level. Work in this area often makes use of techniques derived from source detection, as sources and targets are sometimes construed to both be susceptible to similar entity recognition techniques. We point out in chapter 4 that targets have a different kind of grammatical relationship to opinions from sources—namely, one that associates the polarity of sentiment descriptor words to targets, a fact that can be exploited in polarity detection (chapter 5).

For example, Kim and Hovy (2006) make use of semantic frames and semantic

role labelling to identify sources and targets; their frames come from an existing frame database (FrameNet). However, even within a sentence, sentiment components appear in more varied syntactic contexts than what is offered by simple subcategorization frames.

Algorithms such as Choi et al. (2006), mentioned in the previous section make use of pattern recognition algorithms that will also take into account targets as well as sources. However, like Kim and Hovy (2006), this technique still depends on semantic role labelling from subcategorization frame databases (PropBank).

Work in specifying the problem in a more general way has been recently done by Ruppenhofer et al. (2008). They provide a detailed breakdown of the different semantic and discourse contexts in which sources and targets appear in relation to opinions—contexts beyond variants of subcategorization frames. They do not propose a technique to recognize them.

Recently, a new version of the MPQA was released with target annotations; this effort is described in Wilson (2007).

Our search of the literature found only a small number of efforts in finding opinion targets that did not start from the assumption that the same kinds of techniques could be applied to them as to the sources; we describe these in further detail in chapter 4. We justify why a target-particular technique would be valuable in chapters 4 and 5 and describe a technique to identify them using syntactic dependency graphs. In a nutshell, we hypothesize that the information in the dependency graph path between opinion-bearing syntactic units and target noun phrases can be used to extract features that can improve subjectivity and sentiment classification.

1.3.4 Extracting sources and targets together

Almost all efforts in sentence-level opinion extraction have used the opinion expression itself as the lever by which opinion components are identified; we gave examples of these in the previous sections. Many of these techniques use computationally expensive pattern matching and parsing technologies in order to discover the opinion-bearing sentences themselves.

After reviewing the literature, we were not able to find any work on automatic techniques that discover the sources and targets alone, without the opinions. However, we have an application for which backing slightly away from the sentence level allows us to do just that, using much lighter techniques that open the way for higher volume and higher performance opinion trend analysis. This application is corpus-level trend analysis in the IT business press for social science research in the diffusion of IT innovations. We describe our efforts in this direction in chapter 2 and the application in the next section.

1.4 Application context

Our application for this work is the discovery of networks of influence among opinion leaders in the IT field. We are interested in answering questions about who the leaders in the field are and how their opinion matches the social and economic success of IT innovation. Consequently, it became necessary for us to construct a system (figure 2.1) that finds the expressions in text that refer to an opinion leader's activities in promoting or deprecating a technology.

1.4.1 Corpus-based social science

The "expresses-an-opinion-about" relation is a binary relation between opinion sources and targets. Sources include both people—typically known experts, corporate representatives, and other businesspeople—as well as organizations such as corporations and government bodies. The targets are the innovation terms. Therefore, the use of named-entity recognition in this project only focuses on persons and organizations, as the targets are a fixed list.

1.4.2 Reifying opinion in an application context

A hypothesis implicit in our social science task is that opinion leaders create trends in IT innovation adoption partly by the text that their activities generate in the IT business press. This text has an effect on readers, and these readers act in such such a way that in turn may generate more or less prominence for a given innovation—and may also generate further text.

Some of these text-generating activities include expressions of private states in an opinion source (e.g., "I believe that *Web 2.0 is the future*"). These kinds of expressions suggest a particular ontology of opinion analysis involving discourse relations across various types of clauses [Wilson, 2003, Wilson et al., 2005a]. However, if we are to track the relative adoption of IT innovations, we must take into account the effect of the text on the reader's opinion about these innovations—there are expressions other than those of private states that have an effect on the reader. These can be considered to be "opinionated acts".

Opinionated acts can include things like purchasing and adoption decisions by organizations. For example:

> And like other top suppliers to **Wal-Mart Stores Inc.**, **BP** has been involved in a mandate to affix *radio frequency identification* tags with embedded electronic product codes to its crates and pallets. (ComputerWorld, January 2005)

In this case, both Wal-Mart and BP have expressed implicit approval for radio frequency identification by adopting it from the perspective of the reader's own likelihood of support or adoption of the technology. In this context, we do not directly consider the subjectivity of the opinion source, even though that may be present.

Opinionated acts include things like implications of technology use, not just adoption. We thus define opinion expressions as follows: any expression involving some actor that is likely to affect a reader's own potential to adopt, reject, or speak positively or negatively of a target. This would include "conventional" expressions of private states as well as opinionated acts.

1.5 Organization

The remainder of this document is organized as follows. The next chapter contains our lightweight vector-space machine learning technique for source-target relation detection. The two chapters after that contain a description of our procedure to extract domain-specific grammatical features. Chapter 3 contains our word-level data collection user interface and quality control system, and the procedure we used to annotate a segment of our corpus via crowdsourcing. Chapter 4 uses the data collected through crowdsourcing

as training data for a model that exploits grammatical structure in sentiment-related feature extraction. Finally, we summarize our work in chapter 5, set it in the context of a larger programme of opinion mining research, and propose an application for the data we collect.

Chapter 2

Paragraph-level opinion vicinities

2.1 Introduction

Two problems in sentiment analysis consist of source attribution and target discovery—who has an opinion, and about what? These problems are usually presented in terms of techniques that relate them to the actual opinion expressed. We have a social science application in which the identification of sources and targets over a large volume of text is more important than identifying the actual opinions particularly in experimenting with social science models of opinion trends. Consequently, we are able to use lightweight techniques to identify sources and targets without using resource-intensive techniques to identify opinionated phrases. Most of the work presented in this chapter has appeared as Sayeed et al. [2010b].

In this chapter, we demonstrate an information extraction [Mooney and Bunescu, 2005] approach based in relation mining [Girju et al., 2007] that is effective for this purpose. We describe a technique by which corpus statistics allow us to classify pairs of entities and sentiment analysis targets as instances of an "expresses-an-opinion-about" relation in documents in the IT business press. This genre has the characteristic that many entities and targets are represented within individual sentences and paragraphs. Features based on the frequency counts of query results allow us to train classifiers that allow us to extract "expresses-an-opinion-about" instances, using a very simple annotation strategy to acquire

training examples.

In the IT business press, the opinionated language is different from the newswire text for which many extant sentiment tools were developed. We use an existing sentiment lexicon alongside other non-sentiment-specific measures that adapt resources from newswire-developed sentiment analysis projects without imposing the full complexity of those techniques.

Our definition of "expresses-an-opinion-about" follows immediately. Source A expresses an opinion about target B if an interested third party C's actions towards B may be affected by A's textually recorded actions, in a context where actions have positive or negative weight (e.g. purchasing, promotion, etc.).

2.1.1 Domain-specific sentiment detection

We construct a system that uses named-entity recognition and supervised machine learning via SVMs to automatically discover instances of "expresses-an-opinion-about" as a binary relation at reasonably high accuracy and precision.

The advantage of our approach is that we evade the need for resource-intensive techniques such as sophsticated grammatical models, sequence models, and semantic role labelling [Choi et al., 2006, Kim and Hovy, 2006] by removing the focus on the actual opinion expressed. Then we can use a simple supervised discriminative technique with a joint model of local term frequency information and corpus-wide co-occurrence distributions in order to discover the raw data for opinion trend modelling. The most complex instrument we use from sentiment analysis research on conventional newswire is

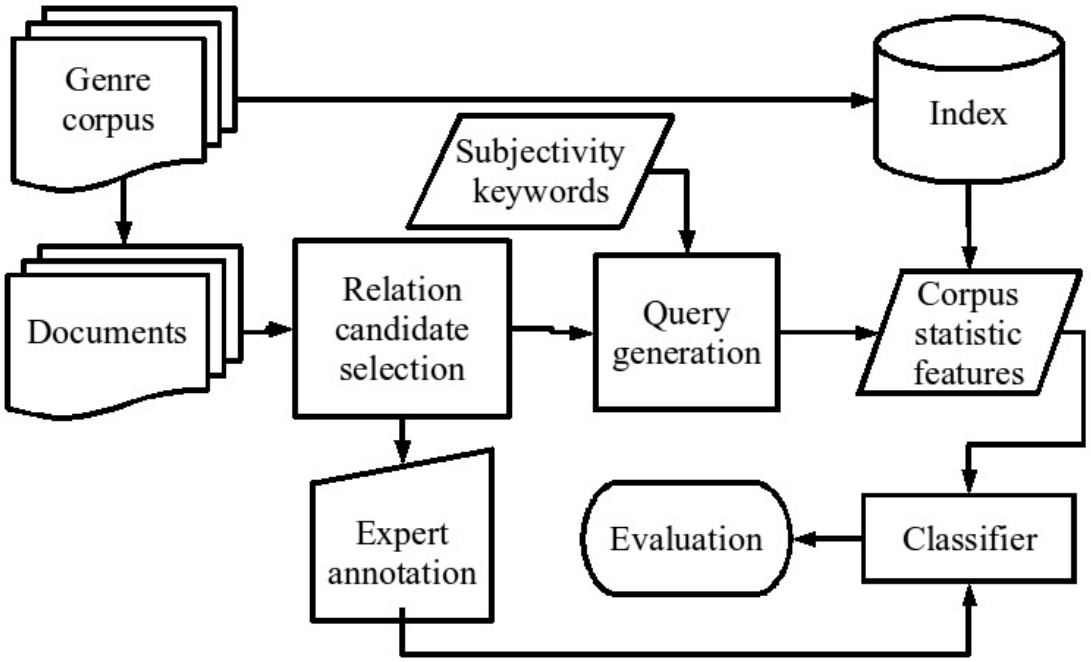


Figure 2.1: Opinion relation classification system.

a sentiment keyword lexicon; furthermore, our techniques allow us to distinguish sentiment keywords that indicate opinion in this domain from keywords that actually indicate that there is no opinion relation between source and target.

While we show that this lightweight technique works well at a paragraph level, it can also be used in conjunction with more resource-intensive techniques used to find "conventional" opinion expressions. The use of topic aspects [Somasundaran and Wiebe, 2009] in conjunction with target names has been associated with an improvement in recall. However, our technique still performs well above the baseline without these improvements.

2.2 Methodology

2.2.1 Article preparation

We have a list of IT innovations on which our opinion leader research effort is most closely focused. This list contains common names that refer to these technologies as well as some alternate names and abbreviations. We selected articles at random from the ComputerWorld IT journal that contained mentions of members of the given list. These direct mentions were tagged in the document as XML entities.

Each article was processed by BBN's IdentiFinder 3.3 [Bikel et al., 1999], a named entity recognition (NER) system that tags named mentions of person and organization entities[1].

The articles were then divided into paragraphs. For each paragraph, we generated candidate relations from the entities and innovations mentioned therein. To generate candidates, we paired every entity in the paragraph with every innovation. Pairs are sometimes generated when an entity is mentioned in multiple ways in the paragraph. We eliminated most of these by removing entities whose mentions were substrings of other mentions. For example, "Microsoft" and "Microsoft Corp." are sometimes found in the same paragraph; we eliminate "Microsoft."

[1] In a separate research effort [Sayeed et al., 2010a], we found that IdentiFinder has a high error rate on IT business press documents, so we built a system to reduce the error *post hoc*. We ran this system over the IdentiFinder annotations.

2.2.2 Annotation

We processed 20 documents containing 157 relations in the manner described in the previous section. Then two domain experts (chosen from the authors) annotated every candidate pair in every document according to the following scheme (illustrated in figure 2.2):

- If the paragraph associated with the candidate pair describes a valid source-target relation, the experts annotated it with `Y`.
- If the paragraph does not actually contain that source-target relation, the experts annotated it with `N`.
- If either the source or the target is misidentified (e.g., errors in named entity recognition), the experts annotated it with `X`.

The Cohen's κ score was 0.6 for two annotators. While this appears to be only moderate agreement, we are still able to achieve good performance in our experiments with this value.

We then selected 75 different documents for each annotator and processed and annotated them as above. At this point we have the instances and the classes to which they belong. We labelled 466 instances of `Y`, 325 instances of `N`, and 280 instances of `X`, for a total of 1071 relations.

Davis says she has especially enjoyed working with the **PowerPad**'s *bluetooth* interfaces to phones and printers. "It's nice getting into new wireless technology," she says. The *bluetooth* capability will allow couriers to transmit data without docking their devices in their trucks.

Source	**Target**	**Class**
Davis	bluetooth	**Y**/N/X
PowerPad	bluetooth	Y/N/**X**

Figure 2.2: Example paragraph annotation exercise.

2.2.3 Feature vector generation

We have four classes of features for every relation instance. Each type of feature consists of counts extracted from an index of 77,227 ComputerWorld articles from January 1988 to June 2008 generated by the University of Massachusetts search engine Indri [Metzler and Croft, 2004]. Each vector is normalized to the unit vector. The index is not stemmed for performance reasons.

The first type of feature consists of simple document frequency statistics for source-target pairs throughout the corpus. The second type consists of document frequency counts of source-target pairs when they are in particularly close proximity to one another. The third type consists of document frequency counts of source target pairs proximate to keywords that reflect subjectivity. The fourth and final type consist of TFIDF scores of vocabulary items in the paragraph containing the putative opinion-holding relation (unigram context features). We use the first three features types to represent the likelihood in the "world" that the source has an opinion about the target and the last feature type to represent the likelihood of the specific paragraph containing an opinion that reflects the

source-target relation.

Positive class	Negative class	System	Prec / Rec / F	Accuracy
Y	N	Random baseline	0.60 / 0.53 / 0.56	0.52
Y	N	Maj.-class (Y) baseline	0.59 / *1.00* / 0.74	0.59
Y	N	Linear kernel	0.70 / 0.73 / 0.72	0.66
Y	N	RBF kernel	**0.72 / 0.76 / 0.75**	**0.69**
Y	N	RBF + alias expand	0.71 / 0.79 / 0.74	0.68
Y	N/X	Random baseline	0.44 / 0.50 / 0.47	0.50
Y	N/X	RBF kernel	0.65 / 0.55 / 0.59	0.67

Table 2.1: Results with all features against majority class and random baselines. All values are mean averages under 10-fold cross validation.

We have a total of 7450 features. Each vector is represented as a sparse array. 806 features represent queries on the Indri index. For all the features, we therefore have 863,226 index queries. We perform the queries in parallel on 25 processors to generate the full feature array, which takes approximately an hour on processors running at 8Ghz. We eliminate all values that are smaller in magnitude than 0.000001 after unit vector normalization.

2.2.3.1 Frequency statistics

There are two simple frequency statistics features generated from Indri queries. The first is the raw frequency counts of within-document co-occurrences of the source and target in the relation. The second is the mean co-occurrence frequency of the source and target per ComputerWorld document.

2.2.3.2 Proximity counts

For every relation, we query Indri to check how often the source and the target appear in the same document in the ComputerWorld corpus within four word ranges: 5, 25, 100, and 500. That is to say, if a source and a target appear within five words of one another, this is included in the five-word proximity feature. This generates four features per relation.

2.2.3.3 Subjectivity keyword proximity counts

We augment the proximity counts feature with a third requirement: that the source and target appear within one of the ranges with a "subjectivity keyword." The keywords are taken from University of Pittsburgh subjectivity lexicon [Wilson et al., 2005b]; the utility of this lexicon is supported in recent work [Somasundaran and Wiebe, 2009].

For performance reasons, we did not use all of the entries in the subjectivity lexicon. Instead, we used a TFIDF-based measure to rank the keywords by their prevalence in the ComputerWorld corpus where the term frequency is defined over the entire corpus. Then we selected 200 keywords with the highest score.

For each keyword, we use the same proximity ranges (5, 25, 100, and 500) in queries to Indri where we obtain counts of each keyword-source-target triple for each range. There are threfore 800 subjectivity keyword features.

2.2.3.4 Word context (unigram) features

For each relation, we take term frequency counts of the paragraph to which the relation belongs. We multiply them by the IDF of the term across the ComputerWorld corpus. This yields 6644 features over all paragraphs.

2.2.4 Machine learning

On these feature vectors, we trained SVM models using the svmlight tool [Joachims, 1999]. We use a radial basis function kernel with an error cost parameter of 100 and a γ of 0.25. We also use a linear kernel with an error cost parameter of 100 because it is straightforwardly possible with a linear kernel to extract the top features from the model generated by svmlight.

2.3 Experiments

We conducted most of our experiments with only the `Y` and `N` classes, discarding all `X`; this restricted most of our results to those assuming correct named entity recognition. `Y` was the positive class for training the svmlight models, and `N` was the negative class. We also performed experiments with `N` and `X` together being the negative class; this represents the condition that we are seeking "expresses-an-opinion-about" even with a

higher named-entity error rate.

We use two baselines. One is a random baseline with uniform probability for the positive and negative classes. The other is a majority-class assigner (`Y` is the majority class).

The best system for the `Y` vs. `N` experiment was subjected to feature ablation. We first systematically removed each of the four feature types individually. The feature type whose removal had the largest effect on performance was removed permanently, and the rest of the features were tested without it. This was done once more, at which point only one feature type was present in the models tested.

2.3.1 Alias expansion

We mentioned in section 2.2.1 that we eliminated duplication in relation instances by using a crude form of coreference detection: we eliminated any entity that was a substring of another entity within that paragraph. However, this potentially removes terms that could produce additional results in Indri queries; the Indri query language permits synonym queries.

For example, the names “Lockheed”, “Lockheed Martin”, and “Lockheed Martin Aeronautics Corp.” refer to the same company, and we would expect that queries involving all three as synonyms might bring up more results than the longest one, which may only be mentioned once in a document. Thus we sought to combine these entity aliases during our feature vector generation by the same substring technique that we used to reduce redundant relations. We searched for aliases at the document level.

We used graph algorithm techniques to deal with special cases where two longer aliases are connected by a shorter alias. An example of this would be "Acme Widget Company", "Acme Company", and "Acme". The first two are not substrings of each other, but they are connected by the third.

2.3.2 Evaluation

All evaluation was performed under 10-fold cross validation, and we report the mean average of all performance metrics (precision, recall, harmonic mean F-measure, and accuracy) across folds.

We define these measures in the standard information retrieval form. If tp represents true positives, tn true negatives, fp false positives, and fn false negatives, then precision is $tp/(tp+fp)$, recall $tp/(tp+fn)$, F-measure (harmonic mean) is $2(prec*rec)/(prec+rec)$, and accuracy is $(tp+tn)/(tp+fp+fn+tn)$.

2.4 Results and discussion

The results of the experiments with all features are listed in table 2.1.

2.4.1 "Perfect" named entity recognition

We achieve best results in the `Y` versus `N` case using the radial basis function kernel. We find improvement in F-measure and accuracy at 19% and 17% respectively. Simply assigning the majority class to all test examples yields a very high recall, by definition, but poor precision and accuracy; hence its relatively high F-measure does not reflect high

Subjectivity	Proximity	Frequency	Unigram	Prec / Rec / F	Accuracy
✓	✓	✓	✓	0.72 / 0.76 / 0.75	0.69
	✓	✓	✓	0.67 / 0.89 / 0.76	0.67
✓		✓	✓	0.71 / 0.77 / 0.73	0.68
✓	✓		✓	0.70 / 0.78 / 0.74	0.67
✓	✓	✓		0.69 / 0.77 / 0.73	0.67
		✓	✓	0.63 / 0.91 / 0.75	0.64
	✓		✓	0.66 / 0.89 / 0.76	0.67
	✓	✓		0.65 / 0.90 / 0.76	0.66
		✓		0.61 / 0.92 / 0.73	0.60
			✓	0.61 / 0.94 / 0.74	0.60

Table 2.2: Feature ablation results for RBF kernel on `Y` vs. `N` case. The first line is the RBF result with all features from table 2.1.

applicability to further processing, as the false positives would amplify errors in our social science application.

The linear kernel has results that are below the RBF kernel for all measures, but are relatively close to the RBF results.

2.4.2 Introducing erroneous named entities

The case of `Y` versus `N` and `X` together unsurprisingly performed worse than the case where named entity errors were eliminated. However, relative to its own random baseline, it performed well, with a 12% and 17% improvement in F-measure and accuracy using the RBF kernel. This suggests that the errors do not introduce enough noise into the system to produce a large decline in performance.

As `X` instances are about 26% of the total and we see a considerable drop in recall, we can say that some of the `X` instances are likely to be similar to valid `Y` instances; indeed, examination of the named entity recognizer's errors suggest that some of the incorrect organizations (e.g. product names) occur in contexts in which valid organizations also occur. However, the precision and accuracy have not fallen nearly as far, so that the quality of the output for further processing is not hurt in proportion to the introduction of noise from the `X` class.

2.4.3 Feature ablation

Table 2.2 contains the result of our feature ablation experiments. Overall, the removal of features causes the SVM models to behave increasingly like a majority class assigner. As we mentioned earlier, higher recall at the expense of precision and accuracy is not an optimal outcome for us even if the F-measure is preserved. In our results, the F-measure values are remarkably stable.

In the first round of feature removal, the subjectivity keyword features have the biggest effect with the largest drop in precision and the largest increase in recall; high-

TFIDF words from a general-purpose subjectivity lexicon allow the model to assign more items to the negative class.

The next round of feature removal shows that the proximity features have the next largest amount of influence on the classifier, as precision drops by 4%. The proximity features are very similar to the subjectivity features in that they too involve queries over windows of limited word sizes; the subjectivity keyword features only differ in that a subjectivity keyword must be within the window as well. That the proximity features are not more important than the subjectivity features, implies that the subjectivity keywords matter to the classifier, even though they are not specific to the IT domain. However, the proximity of sources and targets also matters, even in the absence of the subjectivity keywords.

Finally, we are left with the frequency features and the unigram context features. Either set of features supports a level of performance greater than the random baseline in table 2.1. However, the unigram features allow for slightly better recall than the frequency features without loss of precision, but this may not be very surprising, as there are *many* more unigram features than frequency features. More importantly, however, either of these feature types is sufficient to prevent the classifier from assigning the majority class all of the time, although they come close.

2.4.4 Most discriminative features

The models generated by svmlight under a linear kernel allow for the extraction of feature weights by a script written by svmlight's creator. We divided the instances into a

Feature type	Range	Keyword
Subjectivity	500	agreement
Subjectivity	500	critical
Subjectivity	500	want
Subjectivity	100	will
Subjectivity	100	able
Subjectivity	500	worth
Subjectivity	500	benefit
Subjectivity	100	trying
Subjectivity	500	large
Subjectivity	500	competitive

Table 2.3: The 10 most positive features via a linear kernel in descending order.

single 70%/30% train/test split and trained a classifier with a linear kernel and an error cost parameter of 100, with results similar to those reported under 10-fold cross-validation in table 2.1. We used all features.

Then we were able to extract the 10 most positive (table 2.3) and 10 most negative (table 2.4) features from the model.

Interestingly, all of these are subjectivity keyword features, even the negatively weighted features. The top positive features are often evocative of business language, such

Feature type	Range	Keyword
Subjectivity	500	low
Subjectivity	500	ensure
Subjectivity	25	want
Subjectivity	100	vice
Subjectivity	500	slow
Subjectivity	100	large
Subjectivity	500	ready
Subjectivity	100	actually
Subjectivity	100	ready
Subjectivity	100	against

Table 2.4: The 10 most negative features via a linear kernel in descending order.

as “agreement”, “critical”, and “competitive”. Most of them emerge from queries at the 500-word range, suggesting that their presence in the document itself is evidence that a source is expressing an opinion about a target. That most of them are subjectivity features is reflected in the feature ablation results in the previous section.

It is less clear why “ensure” and “against” should be evidence that a source-target pair is *not* an instance of “expresses-an-opinion-about”. On the other hand, words like “ready” (which appears twice) and “actually” can conceivably reflect situations in the

IT domain that are not matters of opinion. In either case, this demonstrates one of the advantages of our technique, as these are features that actively assist in classifying some relation instances as not expressing sentiment. These features suggest that contrary to what we would expect from general newswire text, "want" in a 25-word window with a source and a target is actually evidence against an "expresses-an-opinion-about" relation in text about IT innovations. This is one example of the phenomenon (ComputerWorld, July 2007):

> But **Klein**, who is director of information services and technology, didn't want IT to become the *blog* police.

In this example, Klein is expressing a desire, but not about the innovation (blogs) in question.

2.4.5 Entity alias expansion

From the results in table 2.1, it appears that expanding the entity aliases in the queries does not improve performance, although it does not hurt precision or accuracy very badly. That suggests that a larger number of aliases introduces noise into the Indri search results.

2.5 Conclusions and future work

2.5.1 Summary

We constructed and evaluated a system that detects at paragraph level whether entities relevant to the IT domain have expressed an opinion about a list of IT innovations

of interest to a larger social science research program. To that end, we used a combination of co-occurrence statistics gleaned from a document indexing tool and TFIDF values from the local term context. Under these novel conditions, we successfully exceeded simple baselines by large margins.

Despite only moderate annotator agreement, we were able to produce results coherent enough to successfully train classifiers and conduct experiments.

Our feature ablation study suggests that all of the feature types played a role in improving the performance of the system over the random and majority-class baselines. However, the subjectivity keyword features from an existing lexicon played the largest role, followed by the proximity and unigram features. Subjectivity keyword features dominated the ranks of feature weights under a linear kernel, and the features most predictive of membership in "expresses-an-opinion-about" are words with semantic significance in the context of the IT business press.

Expanding the source aliases by using substrings does not appear to improve performance.

2.5.2 Application to other domains

We used somewhat naïve statistics in a simple machine learning system in order to implement a form of opinion mining for a particular domain. The most direct linguistic guidance we provided our system were the query ranges and the subjectivity lexicon. The generality of this approach yields the advantage that it can be applied to other domains where there are ways of expressing sentiment unique to those domains outside of newswire

text and product reviews.

2.5.3 Improving the features

Our use of an existing sentiment lexicon opens the door in future work for the use of techniques to bootstrap a larger sentiment lexicon that emphasizes domain-specific language in the expression of opinion, including opinionated acts. In fact, our results suggest that terminology in the existing lexicon that is most prominently weighted in our classifier also tends to be domain-relevant. In a further iteration, we might also improve performance by using terms outside the lexicon that tend to co-occur with terms from the lexicon.

2.5.4 Data generation

Our annotation exercise was a very simple one involving a short reading exercise and the selection of one of three choices per relation instance. This type of exercise is ideally suited to the "crowdsourcing" technique of paying many individuals small amounts of money to perform these simple annotations over the Internet. Previous research [Snow et al., 2008] suggests that we can generate very large datasets very quickly in this way; this is a requirement for expanding to other domains. We expand on this idea in the next chapter.

2.5.5 Scalability

In order to classify on the order of 1000 instances, it took nearly a million queries to the Indri index, which took a little over an hour to do in parallel on 25 processors by calling the Indri query engine afresh at each query. While each query is necessary to generate each feature value, there are a number of optimizations we could implement to accelerate the process. Various types of dynamic programming and caching could be used to handle related queries. One way of scaling up to larger datasets would be to use the MapReduce and cloud computing paradigms on which text processing tools have already been implemented [Moreira et al., 2007].

The intended application for this research was a social science exercise in exploring trends in IT adoption by analysing IT business press corpora. In the end, the perfect discovery of all instances of "expresses-an-opinion-about" is not as important as finding enough reliable data over a large number of documents. This work brings us several steps closer in finding the right combination of features in order to acquire trend-representative data.

As we mentioned in chapter 1, we kept this chapter's work as a separate effort from the work of the following two chapters. In the final chapter of this document, we propose how to integrate the technique of this chapter into a larger polarity-classification pipeline.

Chapter 3

Word-level opinion resource creation

3.1 Introduction

In the previous chapter, we used lightweight techniques to detect the presence of opinionated language at the paragraph level. In this chapter and the next chapter, we change the focus to an even finer-grained level. The contribution that we make in this chapter is a process to perform annotation on a highly subjective, domain-specific task, and to do so in a manner that replicates the judgement of an expert while using unskilled labour from the World Wide Web. The work in this chapter was published as Sayeed et al. [2011].

Crowdsourcing permits us to use a bank of anonymous workers with unknown skill levels to perform complex tasks given a simple breakdown of these tasks with user interface design that hides the full task complexity. Use of these techniques is growing in the areas of computational linguistics and information retrieval, particularly since these fields now rely on the collection of large datasets for use in machine learning. Considering the variety of applications, a variety of datasets is needed, but trained, known workers are an expense in principle that must be furnished for each one. Consequently, crowdsourcing offers a way to collect this data cheaply and quickly [Snow et al., 2008, Sayeed et al., 2010a].

We applied crowdsourcing to perform the fine-grained annotation of our domain corpus. Our user interface design and our annotator quality control process allows these

anonymous workers to perform a highly subjective task in a manner that correlates their collective understanding of the task to our own expert judgements about it. The path to success provides some illustration of the pitfalls inherent in opinion annotation. Our task is: domain and application-specific sentiment classification at the sub-sentence level—at the word level.

3.1.1 Crowdsourcing in sentiment analysis

Paid crowdsourcing is a relatively new trend in computational linguistics. Work exists at the paragraph and document level, and it exists for the Twitter and blog genres [Hsueh et al., 2009].

A key problem in crowdsourcing sentiment analysis is the matter of quality control. A crowdsourced opinion mining task is an attempt to use untrained annotators over a task that is inherently very subjective. It is doubly difficult for specialized domains, since crowdsourcing platforms have no way of directly recruiting domain experts.

Hsueh et al. [2009] present results in quality control over snippets of political blog posts in a task classifying them by sentiment and political alignment. They find that they can use a measurement of annotator noise to eliminate low-quality annotations at this coarse level by reweighting snippet ambiguity scores with noise scores. We demonstrate that we can use a similar annotator quality measure alone to eliminate low-quality annotations on a much finer-grained task.

In this chapter, we describe our work in crowdsourcing fine-grained domain-specific opinion annotation in the following manner. We describe the corpus (section 3.2) that

we use in this work. Then we describe the process that led to the design of our final crowdsourced annotation through direct annotation efforts using trained, local workers and simple, yes-or-no crowdsourced interfaces (section 3.3). This process leads to our indirect crowdsourced user interface design, which we describe in section 3.4. We then describe our data pipeline for populating the user interface including our quality control process (section 3.5). Finally, we describe and discuss our results (section 3.6) and mention some possible future work in this direction (section 3.7).

3.2 Data source

Our corpus for this task is a collection of articles from the IT professional magazine, *Information Week*, from the years 1991 to 2008. This consists of 33K articles of varying lengths including news bulletins, full-length magazine features, and opinion columns. We obtained the articles via an institutional subscription, and reformatted them in XML[1].

Certain IT concepts are particularly significant in the context of the social science application. Our target list consists of 59 IT innovations and concepts. The list includes plurals, common variations, and abbreviations. Examples of IT concepts include "enterprise resource planning" and "customer relationship management". To avoid introducing confounding factors into our results, we only include explicit mentions and omit pronominal coreference.

[1]We will likely be able to provide a sample of sentence data annotated by our process as a resource once we work out documentation and distribution issues.

3.3 Design decisions

Our goal was to generate training data for the task we describe in the next chapter. This primarily required an annotation process that identified words in a sentence that applied to within-sentence target mentions. Our technique was developed after multiple iterations using other approaches which did not succeed in themselves but produced outputs that were amenable to refinement, and so these techniques became part of a larger pipeline. However, the reasons why they did not succeed on their own are illustrative of some of the challenges in both fine-grained domain-specific opinion annotation and in annotation via crowdsourcing under highly subjective conditions.

3.3.1 Direct annotation

We originally intended to stop with the trained local annotation we describe in 3.5.1.1, but collecting opinionated sentences in this corpus turned out to be very slow using annotators we hired directly for this purpose. Despite repeated training rounds, the annotators had a tendency to miss a large number of sentences that the authors found to be relevant. On discussion with the annotators, it turned out that the variable length of the articles made it easy to miss relevant sentences, particularly in the long feature articles likely to contain opinionated language—a kind of "needle-in-a-haystack" problem.

Even worse, however, the annotators were variably conservative about what constituted an opinion. One annotator produced far fewer annotations than the other one—but the majority of her annotations were also annotated by the other one. Discussion with the annotators revealed that one of them simply had a tighter definition of what constituted an

opinion. Attempts to define opinion explicitly for them still led to a situations in which one was far more conservative than the other.

3.3.2 Cascaded crowdsourcing technique

Insofar as we were looking for training data for use in downstream machine learning techniques, getting uniform sentence-by-sentence coverage of the corpus was not necessary. There are 77K sentences in this corpus which mention the relevant IT concepts; even if only a fraction of them mention the IT concepts with opinionated language, we would still have a potentially rich source of training data.

Nevertheless the direct annotation with trained annotators provided data for selecting candidate sentences for a more rapid annotation. We used the process in section 3.5.1.2 and chose the top-ranked sentences. Then we constructed a task design that divided the annotation into two phases. In the first phase, for each candidate sentence, we ask the annotator whether or not the sentence contains opinionated language about the mentioned IT concept. (We permit "unsure" answers.)

In the second phase, for each candidate sentence for which a majority vote of annotators decided that the sentence contained a relevant opinion, we run a second task asking whether particular words (selected as per section 3.5.1.3) were words directly involved in the expression of the opinion.

We tested this process with the 90 top-ranked sentences. Four individuals in our laboratory answered the "yes/no/unsure" question of the first phase. However, when we took their pairwise Cohen's κ score, no two got more than approximately 0.4. We also

took majority votes of each subset of three annotators and found the Cohen's κ between them and the fourth. The highest score was 0.7, but the score was not stable, and we could not trust the results enough to move onto the second phase.

We also ran this first phase through Amazon Mechanical Turk. It turned out that it was far too easy to cheat on this yes/no question, and some workers simply answered "yes" or "no" all the time. Agreement scores of a Turker majority vote vs. one of the authors turned out to yield a Cohen's κ of 0.05—completely unacceptable.

Discussion with the in-laboratory annotators suggested the roots of the problem: it was the same problem as with the direct Atlas.ti annotation we reported in the previous section. It was very difficult for them to agree on what it meant for a sentence to contain an opinion expressed about a particular concept. Opinions about the nature of opinion ranged from very "conservative" to very "liberal." Even explicit definition with examples led annotators to reach very different conclusions. Furthermore, the longer the annotators thought about it, the more confused and uncertain they were about the criterion.

What is an opinion can itself be a matter of opinion. It became clear that without very tight review of annotation and careful task design, asking users an explicit yes/no question about whether a particular concept has a particular opinion mentioned in a particular sentence has the potential to induce overthinking by annotators, despite our variations on the task. The difficulty may also lead to a tendency to cheat. Crowdsourcing allows us to make use of non-expert labour on difficult tasks if we can break the tasks down into simple questions and aggregate non-expert responses, but we needed a somewhat more complex task design in order to eliminate the difficulty of the task and the tendency to cheat. This understanding led to the user interface design and data processing pipeline we describe in

the following sections.

3.4 User interface

Our user interface (figure 3.1) uses a drag-and-drop process through which workers make decisions about whether particular highlighted words within a given sentence reflect an opinion about a particular mentioned IT concept or innovation. The user is presented with a sentence from the corpus surrounded by some before and after context. Underneath the text are four boxes: "No effect on opinion" (none), "Affects opinion positively" (postive), "Affects opinion negatively" (negative), and "Can't tell" (ambiguous).

The worker must drag each highlighted word in the sentence into one of the boxes, as appropriate. If the worker cannot determine the appropriate box for a particular word, she is expected to drag this to the ambiguous box. The worker is presented with detailed instructions which also remind her that most of words in the sentence are not actually likely to be involved in the expression of an opinion about the relevant IT concept[2]. The worker is not permitted to submit the task without dragging all of the highlighted words to one of the boxes. When a word is dragged to a box, the word in context changes colour; the worker can change her mind by clicking an X next to the word in the box.

We used CrowdFlower to manage the task with Amazon Mechanical Turk as its distribution channel[3]. We set CrowdFlower to present three sentences at a time to users.

[2]We discovered when testing the interface that workers can feel obliged to find a opinion about the selected IT concept. We reduced it by explicitly reminding them that most words do not express a relevant opinion and by placing the none box first.

[3]Amazon Mechanical Turk is an online service that manages the distribution of tasks from "requesters"

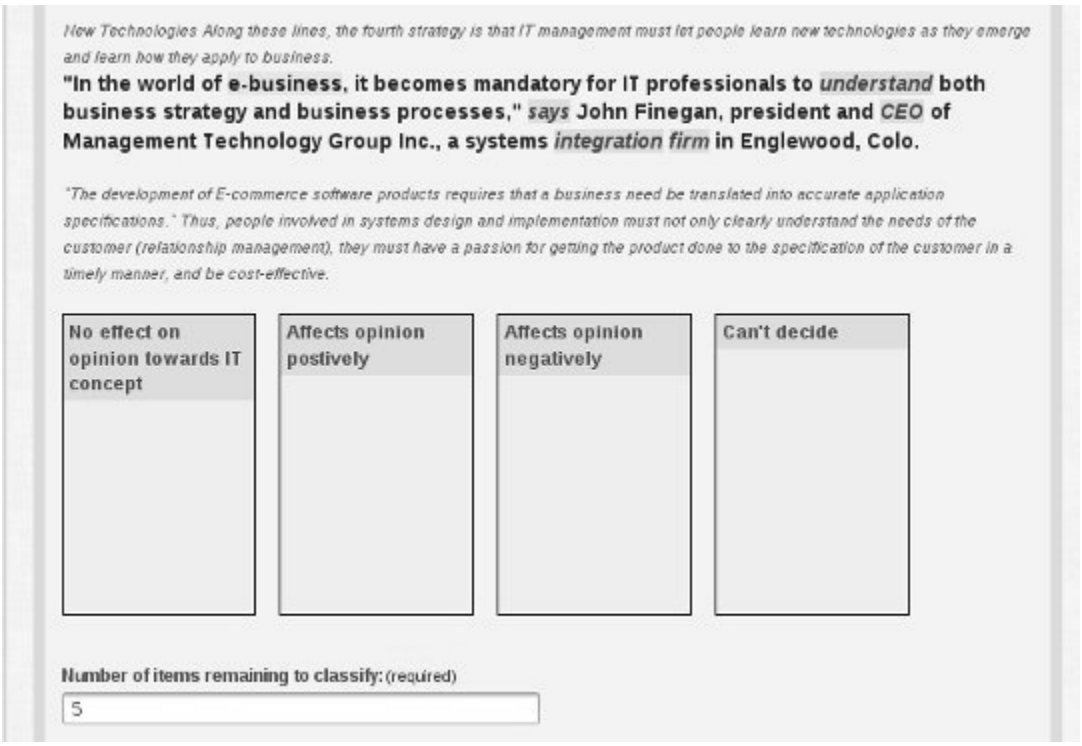


Figure 3.1: A work unit presented in grayscale. “E-business” is the IT concept and would be highlighted in blue. The words in question are highlighted in gray background and turn red after they are dragged to the boxes.

Only users with USA-based IP addresses were permitted to perform the final task.

3.5 Procedure

In this section, we discuss the data processing pipeline (figure 3.3) through which we select candidates for annotations and the crowdsourcing interface we present to the end user for classifying individual words into categories that reflect the effect of the word on the worker.

3.5.1 Data preparation

3.5.1.1 Initial annotation

Two social science undergraduate students were hired to do annotations on *Information Week* with the original intention of doing all the annotations this way. There was a training period where they annotated about 60 documents in sets of 20 in iterative consultation with one of the authors. Then they were given 142 documents to annotate simultaneously in order to assess their agreement after training.

Annotation was performed in Atlas.ti, an annotation tool popular with social science researchers. It was chosen for its familiarity to the social scientists involved in our project and because of their stated preference for using tools that would allow them to share

(task designers) to workers. It returns worker outputs to requesters and disburses requester payments to workers for a fee. CrowdFlower is another online service that provides a task design workbench and a quality control process for crowdsourcing application design and implementation. It feeds tasks through task distribution channels such as Amazon Mechanical Turk.

annotations with colleagues. Atlas.ti has limitations, including the inability to create hierarchical annotations. We overcame these limitations using a special notation to connect related annotations. An annotator highlights a sentence that she believes contains an opinion about a mentioned target on one of the lists. She then highlights the mention of the target and, furthermore, highlights the individual words that express the opinion about the target, using the notation to connect related highlights.

3.5.1.2 Candidate selection

While the use of trained annotators did not produce reliable results (section 3.3) in acceptable time frames, we decided to use the annotations in a process for selecting candidate sentences for crowdsourcing. All 219 sentences that the annotators selected as having opinions about within-sentence IT concepts were concatenated into a single string and converted into a TFIDF unit vector.

We then selected all the sentences that contain IT concept mentions from the entire Information Week corpus using an OpenNLP 1.4.3 model as our sentence-splitter. This produced approximately 77K sentences. Every sentence was converted into a TFIDF unit vector, and we took the cosine similarity of each sentence with the TFIDF vector. We then ranked the sentences by cosine similarity.

3.5.1.3 Selecting highlighted words

We ran every sentence through the Stanford part-of-speech tagger. Words that belonged to open classes such as adjectives and verbs were selected along with certain

The amount of industry attention paid to this new class of integration software speaks volumes about the need to extend the reach of **ERP** systems.

The amount of industry attention paid to this new class of integration software speaks volumes about the need to extend the reach of **ERP** systems.

Figure 3.2: Two highlight groups consisting of the same sentence and concept (ERP) but different non-overlapping sets of candidate words.

closed-class words such as modals and negation words. These candidate words were highlighted in the worker interface.

We did not want to force workers to classify every single word in a sentence, because this would be too tedious. So we instead randomly grouped the highlighted words into non-overlapping sets of six. (Remainders less than five were dropped from the task.) We call these combinations of sentence, six words, and target IT concept a "highlight group" (figure 3.2).

Each highlight group represents a task unit which we present to the worker in our crowdsourcing application. We generated 1000 highlight groups from the top-ranked sentences.

3.5.2 Crowdsourced annotation

3.5.2.1 Training gold

We used CrowdFlower partly because of its automated quality control process. The bedrock of this process is the annotation of a small amount of gold standard data by the

task designers. CrowdFlower randomly selects gold-annotated tasks and presents them to workers amidst other unannotated tasks. Workers are evaluated by the percentage of gold-annotated tasks they perform correctly. The result of a worker performing a task unit is called a "judgement."

Workers are initially presented their gold-annotated tasks without knowing that they are answering a test question. If they get the question wrong, CrowdFlower presents the correct answer to them along with a reason why their answer was an error. They are permitted to write back to the task designer if they disagree with the gold judgement.

This process functions in a manner analogous to the training of a machine-learning system. Furthermore, it permits CrowdFlower to exclude or reject low-quality results. Judgements from a worker who slips below 65% correctness are rated as untrustworthy and not included in the CrowdFlower's results.

We created training gold in the manner recommended by CrowdFlower. We randomly selected 50 highlight groups from the 1000 mentioned in the previous section. We ran these examples through CrowdFlower using the interface we discuss in the next section. Then we used the CrowdFlower gold editor to select 30 highlight groups that contained clear classification decisions where it appeared that the workers were in relative consensus and where we agreed with their decision. Of these, we designated only the clearest-cut classifications as gold, leaving more ambiguous-seeming ones up to the users. For example, in the second highlight group in 3.2, we would designate *software* and *systems* as none and *extend* as positive in the training gold and the remainder as up to the workers. That would be a "minimum effort" to indicate that the worker understands the task the way we do.

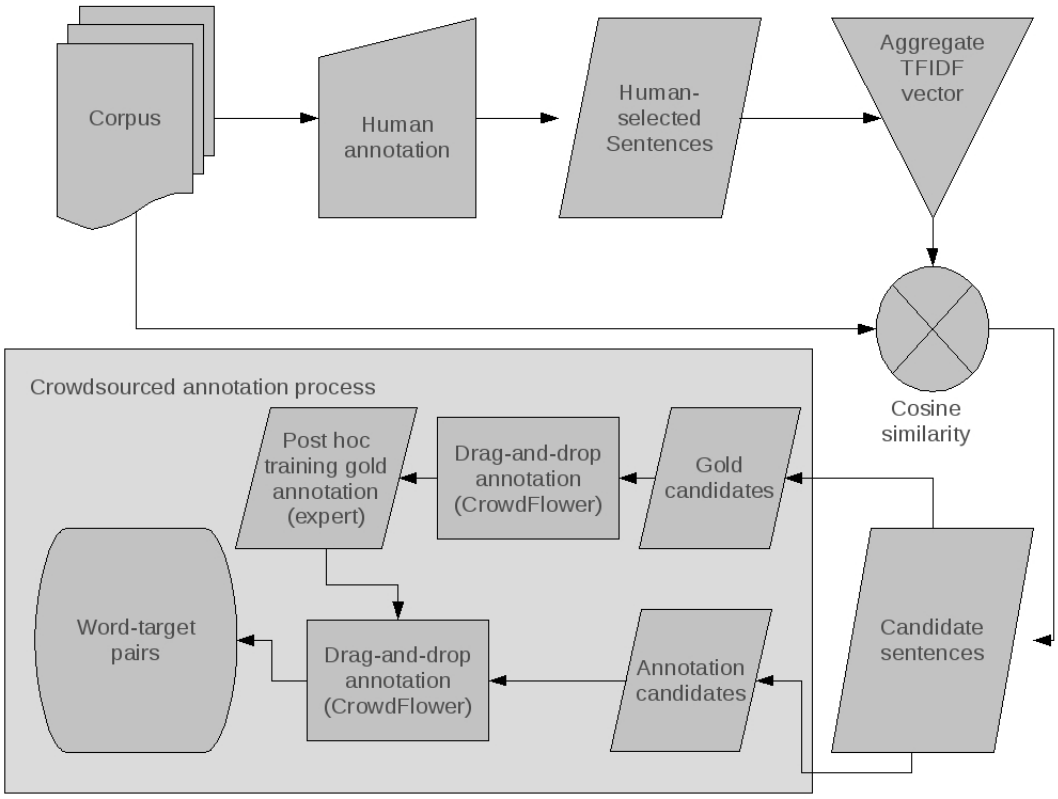


Figure 3.3: Schematic view of pipeline.

Unfortunately, CrowdFlower has some limitations in the way it processes the responses to gold—it is not possible to define a minimum effort precisely. CrowdFlower's setting either allows us to pass workers based on getting at least one item in each class correct or by placing all items in their correct classes. The latter is too strict a criterion for an inherently subjective task. So we accepted the former. We instead applied our minimum effort criterion in some of our experiments as described in section 3.5.3.

3.5.2.2 Full run

We randomly selected another 200 highlight groups and posted them at 12 US cents for each set of three highlight groups, with at least three Mechanical Turk workers seeing each highlight group. The 30 training gold highlight groups were posted along with them. Including CrowdFlower and Amazon fees, the total cost was approximately 60 USD. We permitted only USA-based workers to access the task. Once initiated, the entire task took approximately 24 hours to complete.

3.5.3 Post-processing

3.5.3.1 Aggregation

Each individual worker's ambiguous annotations are converted to none annotations, as the ambiguous box is intended as an outlet for a worker's uncertainty, but we choose to interpret anything that a worker considers too uncertain to be classified as positive or negative as something that is not strongly opinionated under our definitions.

Aggregation is performed by majority vote of the annotators on each word in each highlight group. If no classification obtains more than 50% for a given word, the word is dropped as too ambiguous to be accepted either way as a result. This aggregation has the effect of smoothing out individual annotator differences.

3.5.3.2 Extended quality control

While CrowdFlower provides a first-pass quality control system for selecting annotators who are doing the task in good faith and with some understanding of the instructions,

we wanted particularly to select annotators who would be more likely to be consistent on the most obvious cases without overly constraining them. Even with the same general idea of our intentions, some amount of variation among the annotators is unavoidable; how do we then reject annotations from those workers who pass CrowdFlower's liberal criteria but still do not have an idea of annotation close enough to ours?

Our solution was to score the annotators *post hoc* by their accuracy on our minimum-effort training gold data. Then we progressively dropped the worst n annotators starting from $n = 0$ and measured the quality of the aggregated annotations as per the following section.

3.6 Results

This task can be interpreted in two different ways: as an annotation task and as a retrieval system. Annotator reliability is an issue insofar as it is important that the annotations themselves conform to a predetermined standard. However, for the machine learning task that is downstream in our processing pipeline, obtaining a consistent pattern is more important than conformance to an explicit definition. We can thus interpret the results as being the output of a system whose computational hardware happens to be a crowd of humans rather than silicon, considering that the time of the "run" is comparable to many automated systems; Amazon Mechanical Turk's slogan is "artificial artificial intelligence" for a reason.

Nevertheless, we evaluated our procedure under both interpretations by comparing against our own annotations in order to assess the quality of our collection, aggregation,

and filtering process:

1. **As an annotation task**: we use Cohen's κ between the aggregated and filtered data vs. our annotations in the belief that higher above-chance agreement would imply that the aggregate annotation reflected collective understanding of our definition of sentiment. Considering the inherently subjective nature of this task and the inter-dependencies inherent in within-sentence judgements, Cohen's κ is not a definitive proof of success or failure.

2. **As a retrieval task**: Relative to our own annotations, we use the standard information retrieval measures of precision, recall, and F-measure (harmonic mean) as well as accuracy. We merge positive and negative annotations into a single opinion-bearing class and measure whether we can retrieve opinion-bearing words while minimizing words that are, in context, not opinion-bearing relative to the given target. (We do not merge the classes for agreement-based evaluation as there was not much overlap between positive and negative classifications.) The particular relative difference between precision and recall will suggest whether the workers had a consistent collective understanding of the task.

It should be noted that the MPQA and the JDPA do not report Cohen's κ for subjective text spans partly for the reason we suggest above: the difficulty of assessing objective agreement on a task in which subjectivity is inherent and desirable. There is also a large class imbalance problem. Both these efforts substitute retrieval-based measures into their assessment of agreement.

We annotated a randomly-selected 30 of the 200 highlight groups on our own. Those

30 had 169 annotated words of which 117 were annotated as none, 35 as positive, and 17 as negative. The results of our process are summarized in table 3.1.

Workers excluded	**No. of words lost (of 48)**	**Prec/Rec/F**	**Acc**	**Cohen's κ**	**Score threshold**
(prior polarity)	N/A	0.87 / 0.38 / 0.53	0.79	*-0.26*	N/A
0	0	0.64 / 0.71 / 0.67	0.79	0.48	0.333
1	0	0.64 / 0.71 / 0.67	0.79	0.48	0.476
3	0	0.66 / 0.73 / 0.69	0.80	0.51	0.560
5	0	0.69 / 0.73 / 0.71	0.81	0.53	0.674
7	2	0.81 / 0.76 / **0.79**	0.86	**0.65**	0.714
10	9	**0.85** / 0.74 / **0.79**	**0.88**	0.54	0.776
12	11	0.68 / 0.68 / 0.68	0.82	0.20	0.820

Table 3.1: Results by number of workers excluded from the task. The prior polarity baseline comes from a lexicon by Wilson et al. [2005b] that is not specific to the IT domain.

In the 30 highlight groups, there were 155 total words for which a majority consensus (>50%) was reached. 48 words were determined by us in our own annotation to have opinion weight (positive or negative). There are only 22 annotators who passed CrowdFlower's quality control.

The stringent filter on workers based on their accuracy on our minimum-effort gold annotations has a remarkable effect on the results. As we exclude workers, the F-measure and the Cohen's κ appear to rise, up to a point. By definition, each exclusion raises the

threshold score for acceptance. As we cross the 80% threshold, the performance of the system drops noticeably, as the smoothing effect of voting is lost. Opinion-bearing words also reduce in number as the threshold rises as some highlight groups simply have no one voting for them. We achieve our best result in terms of Cohen's κ on dropping the 7 lowest workers. We achieve our highest precision and accuracy after dropping the 10 lowest workers.

Between the 7th and 10th underperforming annotator, we find that precision starts to exceed recall, possibly due to the loss of retrievable words as some highlight groups lose all their annotators. Lost words can be recovered in another round of annotation.

3.6.1 Discussion

We have been able to show that crowdsourcing a very fine-grained, domain-specific sentiment analysis task with a nonstandard, application-specific definition of sentiment is possible with careful user interface design and mutliple layers of quality control. Our techniques succeed on two different interpretations of the evaluation measure, and we can reclaim any lost words by re-running the task. We used an elaborate processing pipeline before and after annotation in order to accomplish this. In this section, we discuss some aspects of the pipeline that led to the success of this technique.

3.6.2 Quality

There are three major aspects of our procedure that directly affect the quality of our results: the first-pass quality control in CrowdFlower, the majority-vote aggregation, and

the stringent *post hoc* filtering of workers. These interact in particular ways.

The first-pass quality control interacts with the stringent filter in that even if it were possible to have run the stringent filter on CrowdFlower itself, it would probably not have been a good idea. Although we intended the stringent filter to be a minimum effort, it would have rejected workers too quickly. It is technically possible to implement the stringent filtering directly without the CrowdFlower built-in control, but that would have entailed spending an unpredictable amount more money paying for additional unwanted annotations from workers.

Furthermore, the majority-vote aggregation requires that there not be too few annotators; our results show that filtering the workers too aggressively harms the aggregation's smoothing effect. The lesson we take from this is that it can be beneficial to accept some amount of "bad" with the "good" in implementing a very subjective crowdsourcing task.

3.7 Future work

Foremost among the avenues for future work is experimentation with other vote aggregration and *post hoc* filtering schemes. For example, one type of experiment could be the reweighting of votes by annotator quality rather than the wholesale dropping of annotators. Another could involve the use of general-purpose sentiment analysis lexica to bias the vote aggregation in the manner of work in sentiment domain transfer [Tan et al., 2007].

This work also points to the potential for crowdsourcing in computational linguistics applications beyond opinion mining. Our task is a sentiment-specific instance of a large

class of syntactic relatedness problems that may suitable for crowdsourcing. One practical application would be in obtaining training data for coreference detection. Entity coreference detection [Haghighi and Klein, 2009] is a longstanding problem in computational linguistics where a piece of text may have multiple mentions of the same context in different form. This frequently occurs within a sentence (e.g., "The man rode his beautiful *stallion* too long for the *horse* to avoid feeling that *he* was tired"). We could deploy a similar system to find training data for these kinds of links, including for other languages that have a presence on services like Amazon Mechanical Turk.

Another future direction may be in the establishment of empirical support for theories about syntactic structure. Some language with large online presences, such as Russian, permit a great degree of word order freedom, to the point where related sentence constituents may be separated by a number of other words. For example, these languages may permit a verb's object to become separated from the verb and appear outside a subordinate clause [Sayeed and Weinberg, 2009]. This type of interface could be used to identify these cases; the user could drag the constituents that belong to a verb into one of the boxes.

The next chapter focuses on a machine-learning application for the data collected through the user interface described in this chapter.

Chapter 4

Word-level opinion mining via syntactic relatedness tries

In this chapter, we provide a framework for mining opinionated language at a word level. Unlike many previous approaches that have ignored the role of syntax, our approach models sentiment as a latent variable embedded in a syntactically derived factor graph. This approach allows us to leverage features hitherto only used indirectly: direct learning over grammatical structures. We demonstrate that it is now possible to use machine learning to acquire grammatical patterns that contain features that connect opinion-expressing words and target entities. We do this using a data structure we call a "syntactic relatedness trie" (SRT) that allows us to consider the problem in terms of labelling the parts of a structure, rather than labelling a sequence.

Early approaches to opinion mining focused on simply characterizing whether a span of text is sentiment bearing or not [Pang et al., 2002, Turney and Littman, 2003], but modern approaches seek to identify finer-grained aspects of sentiment, which adds additional complexity to a problem already burdened by often subjective distinctions.

We explore how the syntactic connections between words affects how opinions are expressed. Such an approach requires a formalism for analyzing sentiment at a very fine-grained level, which we describe in Section 4.5. Because of a dearth of resources for this detailed task, we also develop new techniques for labeling word-level, syntactically informed sentiment, which we describe in Section 4.4. We then embed our linguistic

formalism, derived from dependency parses, within a probabilistic framework where opinion serves as a latent variable within a SRT encoded as a factor graph (Section 4.6). Finally, we demonstrate that our approach overcomes the technical barrier in detecting opinion-relevant features from fine-grained syntactic structure in Section 4.7.

4.1 Related work

There is recent work on using grammatical features in sentiment analysis and opinion mining, but most of this work has been focused on extracting binary features for use in bag-of-features approaches to sequence labelling and classification. For example, Jakob and Gurevych [2010] perform target identification from known opinion spans within a sentence using paths extracted from dependency parses. They use the single shortest path from a target expression to an opinion span as a binary feature in a conditional random field (CRF) model that labels words with their status as a word that belongs to a target expression. This is an indirect use of grammar that does not provide grammatical features for downstream processing.

Jakob and Gurevych's approach is based on a technique proposed by Zhuang et al. [2006] in the movie review domain. They use dependency graph shortest paths to identify relations between movie feature keywords ("script", "directing") and a known list of opinion keywords by using paths extracted directly from training data as templates rather than using machine learning to develop models that allow for path variability. Zhuang et al.'s approach does not directly handle complex cases of implicit opinion ("Please take this movie away, quickly!") and is focused on summarizing the highlights of movie reviews;

this may not be fully appropriate for domains and genres that have strong implicit and pragmatic components to opinion.

Nakagawa et al. [2010] use a procedure that is superficially similar to ours in that they apply factor graph modeling to their own dependency grammar-based formalism. However, their application is fundamentally different: sentiment polarity classification of known expressions. They assume full sentences preselected for subjectivity and then use the syntactic structures of these sentences to propagate positive and negative polarity until a classification for the whole sentence can be found. Although bag-of-features approaches have been relatively successful [Wilson, 2007] at this kind of task, Nakagawa et al. are still able to achieve a significant improvement over existing work.

4.2 Our contribution

Little work has been done to perform target and opinion-expression extraction jointly, especially in a way that extracts features for downstream processing through techniques such as that of Nakagawa et al. This is not surprising: it is challenging to separate the parts of the grammar that matter from the parts that do not. This requires an investment in novel techniques and formalisms that can handle grammatical structure robustly. Until recently, the lack of such techniques made it difficult to rely on grammatical structures in information extraction applications. Recent work has shown the value of access to fully-connected grammatical structures [Moilanen and Pulman, 2007] in sentiment analysis.

This work exploits advances in machine learning techniques to create the framework for discriminating between relevant grammatical structure and irrelevant structure in the

context of finding the syntactic relations that tie opinion targets and their associated opinion expressions together.

4.3 Feasibility and necessity

Our approach uses a machine learning technique over a rich grammatical representation in order to develop a classifier for future instances. There are two questions we must answer in estimating whether or not this is an experimental process that might bear fruit: whether it solves an unsolved problem, and whether it does so in a technically feasible manner. The problem that we want to solve is the identification of opinion-bearing language; most importantly, we want to find as closely as possible the locations of words that confer sentiment on a target. This is necessary to solve because it is the opinion-bearing language that allows us to classify opinion polarity. The question of the feasibility of our approach is particularly interesting considering the technical challenges of using extensive grammatical formalism in a situation in which there may potentially be much rich linguistic variation.

We assess the potential value of this technique via a preliminary study of the MPQA corpus, wherein we describe what it means to have a grammatical context that is sufficiently complex to warrant the use of a grammar-based technique and determine the frequency of these contexts. We discuss this in the context of work on opinion polarity detection. We then discuss existing work in information extraction and sentiment analysis that uses rich grammatical formalism, suggesting a general idea about the feasibility of rich tehcniques.

4.3.1 Resources: MPQA adaptation and study

The MPQA 2.0 corpus contains "attitude" annotations, which are opinion-target pairs. There are a number of attitude types; we focused on "sentiment-pos" and "sentiment-neg" attitudes (using the MPQA's own terms). We investigated the following question: were there proportionately enough examples of sentiment word-target relations that were grammatically "complex"? That is, we wanted to know how many examples there were that met the following set of heuristics wherein the target-opinion word relation meets the following criteria:

- It is **not** mediated directly by a copula verb (as in "Ike is wonderful").
- Sentiment is **not** expressed by an adjective or verb directly before or after the target ("wonderful Ike").
- Sentiment is **not** expressed by a noun or verb followed by a preposition directly before the target ("I trust in Ike").
- Any negation is involved in the expression of sentiment. ("I do not consider Ike trustworthy.")

We discuss the nature and significance of complex sentiment contexts in section 4.3.1.1.

Human judgements allow us to use the above criteria to divide a sample of 150 "sentiment-pos" and "sentiment-neg" annotations from 100 randomly selected MPQA sentences into three classes: Y, N, and Q. Y class annotations conform to the above criteria, while N do not. Q annotations are those that were questionable either due to inconsistencies

in the MPQA annotation or due to the human annotator's difficulty in deciding on the correct annotation.

This exercise was not intended to generate training data directly, only to assess the applicability of the technique. Consequently, there was only one annotator.

Of the 150 annotations, 63 were N, 70 were Y and 17 were Q, based on the annotator's judgements. This suggests that there is a case for the discovery of complex syntactic patterns beyond adjacent words, considering the large proportion of Y cases.

4.3.1.1 Grammatical complexity in opinion-bearing contexts

Consider the following sentence from the MPQA:

> Until the Pentagon sorts out the legal issues, criticism from Europe is likely to grow.

The MPQA attitude annotation holds that "criticism" represents negative sentiment towards the Pentagon. A simple bag-of-words technique would be unlikely to allow us to distinguish between the Pentagon and Europe as the target of "criticism". A bag-of-words approach biased for string proximity of the sentiment words to a target would tend to suggest that Europe is the target of negative sentiment, being closer to "criticism". If we do some crude syntactic chunking of that sentence into phrases or clauses, "criticism" would still be more proximate to the string "Europe" than to "Pentagon".

This is effectively what it means for an expression of a sentimental attitude to be grammatically complex: that, in order to find it, we need techniques that bound a given sentiment-bearing expression in ways that are not strongly biased towards string proximity

of it to its target.

Another even more pathological example of a complex context in the MPQA is this one:

> The Kyoto pact, which Bush Thursday called "an unsound international treaty", commits countries to cut their emissions of greenhouse gases to below 1990 levels by a timetable of 2008-2012.

The word "unsound" is in a quote inside a relative clause, but it confers negative sentiment upon "pact." We ran this sentence through the Stanford dependency parser, and it transitively connects "unsound" to "pact", via at least one path that does not also pass through "Bush".

4.3.2 Advantages in opinion polarity detection

Grammatical information will not only permit us to identify target-opinion word pairs, but the knowledge of which words specifically relate to which targets will allow us to classify the polarity of the opinion towards the target with greater accuracy.

In the first chapter, we mention existing work in the determination of opinion targets and, in particular, opinions along with their targets. This kind of work is able to find the opinionated text with relatively high accuracy. Additional work on opinion polarity detection [Wilson, 2007] suggests that we can classify opinions by polarity also with high accuracy—given opinion text with boundaries annotated in advance in the MPQA.

While we have found no published results on this to date, private communication with Yejin Choi suggests that attempting to find opinion polarities from automatically detected

opinion text allows the errors in each to be compounded, leading to poor performance. We were not able to find work that directly pipelines automatically detected opinion text into a polarity classifier, including the extensive work by Wilson.

One reason why this may be the case is suggested by Wilson's evaluation technique: she does not commit her identification of opinion-bearing text to match the exact bounds described in the MPQA ground truth—an overlap of detected text with ground truth is considered a success. But precise bounds matter:

> The slain militants owing allegiance to Lashker-e-Toiba have been identified as Abu Saifullah, Abu Talha and Abu Huraira.

The MPQA ground truth holds that "owing allegience to" is a positive sentiment about Lashkar-e-Toiba. A single additional word with negative connotations such as "militants", however, could potentially affect how a polarity classifier classifies this sentence.

The technique we present, however, goes in the opposite direction: finding relevant individual opinion words, rather than whole phrases, thereby avoiding a risk of problematic overlap. As our work searches for opinion words with respect to the targets of an opinion—and as a by-product, produces dependency graph information, including negations—we expect that this technique will improve polarity classification, as we discuss in chapter 5.

4.3.3 Feasibility: rich grammatical information

The use of rich grammatical information in this kind of task raises questions of brittleness and data sparsity to which any grammar-based approach is nowadays subjected. However, there exist tasks of similar complexity for which these kinds of approaches have

been shown to be successful, including in the area of information extraction.

For example, Haghighi and Klein [2009] developed a heavily syntactic approach to coreference resolution, using a small set of hand-built constraints over parse trees and automatically extracted patterns. Their system outperfomed state-of-the-art unsupervised systems and was competitive with supervised systems. Coreference is well-studied in the theoretical linguistics literature, which appears to have allowed Haghighi and Klein to compete with more data-driven techniques. Insofar as sentiment relations are not as well studied in a theoretical context, we choose to back off to a more machine-learning oriented framework that uses grammatical structures as data.

Kim and Hovy [2005] have shown the feasibility of this approach for a related application in sentiment detection. They develop a system that uses machine learning techniques over features that include paths derived from parse trees in order to detect opinion sources. These syntactic features contribute in their highest-performing systems, as measured by accuracy.

The work by Kim and Hovy suggests that machine-learning techniques can work well over detailed syntactic features. Our work differs from Kim and Hovy in that we do not treat an entire path as a single feature function. This has resulted for them in a data sparseness problem that they resolve via boosting. Instead, we model a sequence of labels across hetergeneous paths, cutting down on the number of possible syntactic features and increasing the data generated per relation pair instance. The rest of this chapter explores the characteristics of a model built on this premise.

4.4 Data source

4.4.1 Existing resources

We have a downstream application for this task which involves acquiring patterns in the distribution of opinion-bearing words and targets using machine learning (ML) techniques. Supervised ML techniques require gold standard data annotated in advance.

Ordinarily, we would consider the Multi-Perspective Question-Answering (MPQA) newswire corpus [Wilson and Wiebe, 2005] and the J. D. Power & Associates (JDPA) automotive review blog post [Kessler et al., 2010] corpus to be appropriate because both contain sub-sentence annotations of sentiment-bearing language as text spans. In some cases, they also include links to within-sentence targets. This is an example of an MPQA annotation:

> That was the moment at which the fabric of compassion tore, and worlds cracked apart; when **the contrast and conflict of civilisational values** became so great as to *remove any sense of common ground* - even on which to do battle.

The italicized portion is intended to reflect a negative sentiment about the bolded portion. However, while it is the case that the whole italicized phrase represents a negative sentiment, "remove" appears to represent far more of the negativity than "common" and "ground". While there are techniques that depend on access to entire phrases, our project is to identify sentiment spans at the length of a single word. We are also attempting to succeed at a particular application whose metasubjective perspective is different from that used to

annotate the MPQA or JDPA.

For both these reasons and the reasons mentioned in section 4.3.1, we are instead using our own resource for the work described in the following sections.

4.4.2 Crowdsourced annotation process

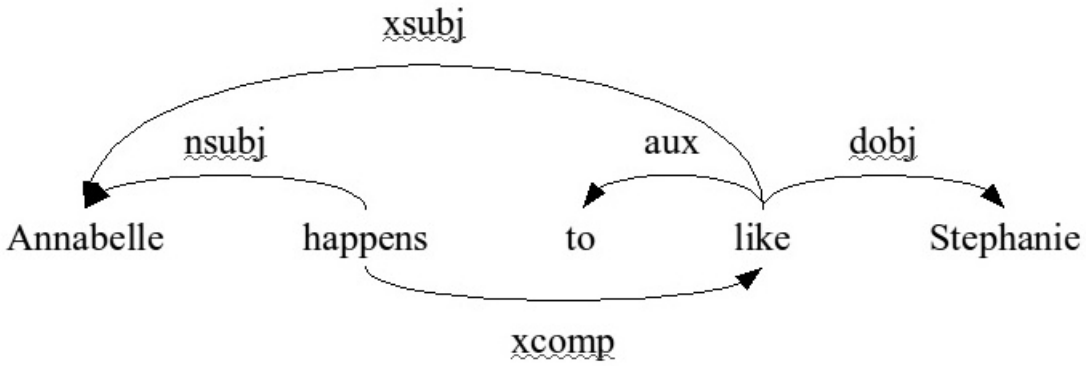
As we mentioned in the previous section, instead of using existing resources, we constructed our own, and we did so using the crowdsourcing process described in chapter 3.

Of the top-ranked 1000 highlight groups mentioned in chapter 3, 700 were used for results in this paper. The total cost of this exercise was appoximately 250 USD, which includes the fees charged by Amazon and CrowdFlower. These last highlight groups were converted to SRTs and divided into training and testing groups, 465 and 196 SRTs respectively, with a small number lost to fatal errors in the Stanford parser.

4.5 Syntactic relatedness tries

Now that we are able to annotate opinion-bearing tuples, we must find a way to represent the within-sentence relationship between target entities and opinion words. Because we are modeling the syntactic connections between the target mentions and opinion words, our representation must retain the syntactic information in the sentence. In this section, we develop the formalism we call a syntactic relatedness trie to encode this information, which we extract from a dependency parse of a sentence.

Figure 4.1: Dependency parse example



4.5.1 Dependency Parse Trees

Dependency parsing is a technique for recovering the grammatical structure of a sentence in terms of binary dependencies between words. A comprehensive overview of dependency parsing is available from Kübler et al. [Kübler et al., 2009]. A dependency parser generates a dependency graph given a sentence. For our purposes, a dependency graph is a directed graph whose nodes are the words in a sentence and whose labeled, directed edges are syntactic relationships between those words.

We use the Stanford Parser [de Marneffe and Manning, 2008]. For the sentence "Annabelle happens to like Stephanie", the parser produces a dependency graph as in Figure 4.1. We use the graph to provide the features over which our system learns the syntactic connection between targets and relevant opinion words.

The Stanford Parser also provides us with linguistic features on the nodes that we use in our training data. It provides us with the parts of speech (POS) tags for each node (word) in the graph. It also provides us with the labels for the directed edges, which are the dependency relationships between words.

A pair of words are connected syntactically in an indirect manner more often than not: there are usually multiple intervening items. We call these (ordered) intervening items a "path." There are often multiple paths connecting a pair of two arbitrary words in a sentence. In the next section, we encode these possibilities.

4.5.2 Encoding Dependencies in an SRT

Syntactic relatedness tries (SRT) enable us to encode the connections between a single linguistic object of interest—in this application, a word and its features—and a set of one or more related objects in the representation. More concretely, a SRT is a datastructure consisting of nodes and edges. No node has more than one incoming edge, but each node may have zero or more outgoing edges, and therefore it forms a tree.

So far, this description is very similar to the definition of a dependency parse, from which the SRT is derived. The key difference is that while a token only appears once as a node in an dependency parse, an SRT can contain multiple nodes that originate from the same token. This encodes the multiple possible connections between a target of an opinion and the words that encode that opinion. We illustrated an example of a sentence where this is important in section 4.3.1.1 ("The Kyoto pact, which Bush Thursday called 'an unsound international treaty', commits countries to cut their emissions of greenhouse gases to below 1990 levels by a timetable of 2008-2012").

For our purposes, the object of interest is the opinion target, so it becomes (by definition) the root node. Each directed edge in the SRT corresponds to a grammatical relationship between words and is labeled with that relationship. We use the notation $a \xrightarrow{\mathrm{R}}$

b to signify that node a has the grammatical relationship R with b. We call R a "role". We say in this case that node b is a descendent of node a with the role R. The root node of an SRT has an implied incoming edge with the role "start" which we omit in the notation. For simplicity, we include the incoming edge labels as part of the node's data structure and use it as a node feature in our machine learning procedure.

We call this tree a trie because it is constructed as a suffix tree into which are inserted *all* paths that begin with a given target and end with an opinionated word. Multiple paths may share elements *en route* between the target and an opinionated word through a dependency graph. Insofar as these elements are shared from the root onwards, these overlapping paths reflect evidence for the relevance of multiple leaves; they also contain features to classify the relationship between the leaves and the root.

In the remainder of this section, we provide an example of how we can construct an SRT from a dependency parse. In addition, we add an additional feature of an SRT: validity. Each node contais an binary variable that permit us to discriminate between relevant features in the tree and irrelevant features.

4.5.3 Validity labels

Our goal is to discriminate between parts of the structure that are relevant to target-opinion word relations and those that are not. We use the term valid for relevant parts of the structure and invalid otherwise. Insofar as SRTs are graph structures, valid and invalid structures are components of the graph. Consequently, we label the nodes of a valid component as valid and vice versa. Validity labels are therefore a feature of SRT

nodes.

Given that the leaves of an SRT are words that potentially express an opinion about the root, the validity label of a node reflects its contribution to that relationship in the grammar.

4.5.4 Dependency graphs and paths

Now we describe the manner in which validity labels are applied to nodes in a dependency path in order to help us discriminate between nodes that lead from a target mention to an opinion word and those that do not. In figure 4.1, the opinion-bearing relation is not complex, if we change the direction of the arrows to be the same along a path[1]. We know that in "Annabelle happens to like Stephanie", "like" is an opinion about "Stephanie":

Stephanie $\xrightarrow{\text{dobj}}$ like

There is also a relationship between "like" and "Annabelle", but it is much longer. Nevertheless, it contains other kinds of relationships along two possible routes:

Annabelle $\xrightarrow{\text{xsubj}}$ like

Annabelle $\xrightarrow{\text{nsubj}}$ happens $\xrightarrow{\text{xcomp}}$ like

Both of these relationships contain some kind of subject role, while the path for "Stephanie" contains a direct object role. But "Stephanie" is the only target of "like"—it is a valid, though short, path. Therefore, any path between "Annabelle" and "like" is not a valid path. We can encode this in the structure by labeling the nodes.

Stephanie:valid $\xrightarrow{\text{dobj}}$ like:valid

Annabelle:invalid $\xrightarrow{\text{xsubj}}$ like:invalid

[1] We do change the arrows in our procedure, instead ignoring the direction of the original dependency parse. However, arrow directionality in the dependency graph is a potential feature for further experimentation.

Annabelle:invalid $\xrightarrow{\text{nsubj}}$ happens:invalid $\xrightarrow{\text{xcomp}}$ like:invalid

This provides the basis for the construction of the SRT, as we describe in the next section.

4.5.5 SRT construction

The distinctions between valid and invalid paths between target entities and opinion-bearing words can be quite fine, and there can be considerable overlapping areas between two or more paths with conflicting validity labels. For example, consider this sentence from the MPQA:

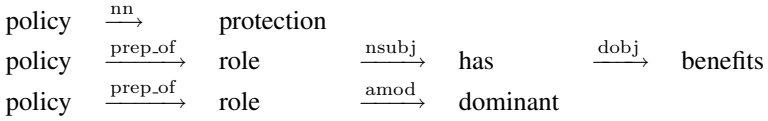
> The *dominant* role of the European climate *protection* **policy** has *benefits* for our economy.

In this sentence, "policy" is connected to multiple sentiment-bearing words[2]. Then we can represent these relationships using paths through the graph as in Figure 4.2(a). We only choose a sample of paths; our representation is intended to encode *all* paths to avoid missing features that may exist on a more indirect path.

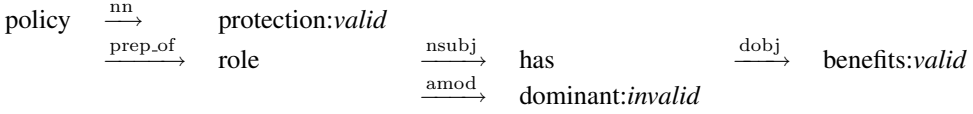
Suppose that an annotator decides that "protection" and "benefits" are directly expressing a positive or negative opinion about the policy, but "dominant" is ambiguous (as it does contain negative connotations). The "protection" and "benefits" paths are valid, and the "dominant" path is invalid. However, there is considerable overlap between the

[2]We do not include the full graph of this sentence for typographical reasons, but a representation can be obtained from the parser web interface at http://nlp.stanford.edu:8080/parser/index.jsp. We use the "collapsed" parsing mode. Our IT examples are usually much larger.

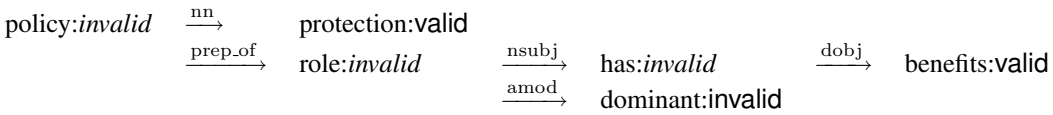
Figure 4.2: An example SRT construction.



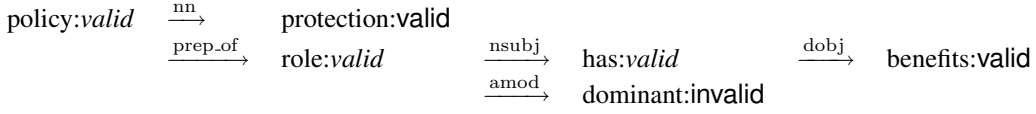
(a) Separate paths.



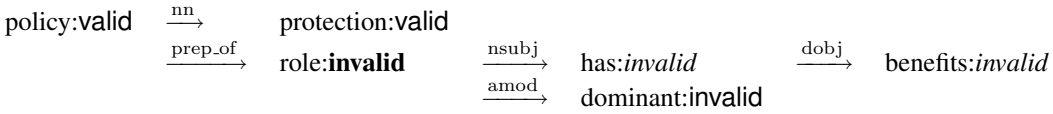
(b) Insertion into trie and labeling of leaves.



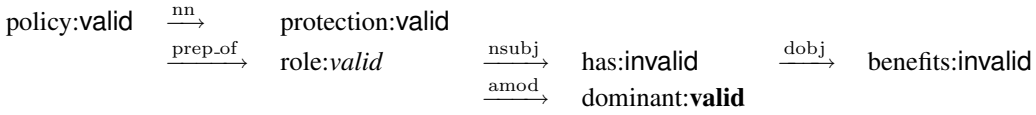
(c) Labeling of root and internal nodes.



(d) Propagation of valid labels.



(e) Single setting change from valid to invalid in (d) with propagation.



(f) Single setting change from invalid to valid in (e) with propagation.

"dominant" path and the "benefits" path that may potentially confound machine learning algorithms. That is the motivation for combining them into a trie structure and labelling them in such a way that the path remains valid until there is *no* path element that leads to a valid leaf. We return to this motivation in section 4.5.6.

In other words, we want the path elements common to a `valid` path and an `invalid` path to reinforce validity, and only the transition from `valid` to `invalid` to be learned by the classifier.

We enforce this requirement in the following manner:

1. We collect every path p that starts with a given target and ends with an opinionated word. In the "policy" example, this is figure 4.2(a).

2. We create an empty trie T.

3. For every p, we insert it into T, with individual path elements acting as nodes of T. The last path element is labelled with the corresponding path validity class. Our example trie looks like figure 4.2(b).

4. Label every node in T except for the final path elements with `invalid`. The example is in 4.2(c).

5. For each `valid` node, label its parent `valid` successively until the root of T. The example is in 4.2(d).

This is equivalent to finding the depth-first search tree of the dependency graph and pruning any branches that do not end in a relevantly opinionated word. Then we can apply the node-labelling scheme as above.

4.5.6 Overlapping paths

Do paths with contradictory labels overlap significantly in the data? We calculated the extent of the overlap in the held-out testing data. In all of the SRTs in the held-out data,

there are a total of 6,669 nodes with validity labels. Of these, 2,215 are leaves, and cannot have any conflict by definition, as they are the crowdsourcing-labelled words themselves.

This leaves 4,454 internal nodes. We calculated that 962 of these are nodes that have multiple descendent leaves that include both valid and invalid nodes—that is, they are valid nodes in relation to paths that lead to some candidate opinionated words but not others. So approximately 22% of internal nodes would produce conflicting evidence if they were not merged.

However, our dependency paths lead from the root (target mention) to the leaves (candidate opinionated words). All these conflicts therefore necessarily congregate at the beginning of paths and interfere disproportionately early on in the sequential labelling of a path with validity labels. Any target mention with both valid and invalid candidate words would itself have a conflict if the paths were kept separate. Merging the paths via the SRT construction ensures that all these nodes are labelled valid, preserving their potential to lead to a relevantly opinionated word.

4.5.7 Invariant

The purpose of an SRT is to provide a framework to discriminate between items that are *en route* from a target to an opinion-relevant word and those that are not, such as "happens" in the Annabelle/Stephanie example. This is encoded by the valid and invalid labels. Anything that follows a node with an invalid label is by definition not reachable from the root of the tree. Consequently, any valid node that follows an invalid node is not reachable along a path and is actually invalid itself. We specify this directly as an invariant

on the data structure:

Invariant: no node descending from a node labelled invalid has a valid label.

This specifies that valid labels spread out from the root of the SRT. It has implications for sampling and inference. Every individual label setting chosen by a sampler during training will affect neighbouring labels and could therefore affect large portions of the trie. A valid label switched to invalid will require all the descendents of that particular node to switch to invalid as well as in figure 4.2(e). Similarly, an invalid label switched to valid will require all of the ancestors of that node to switch to valid as in 4.2(f).

Furthermore, the initial setting must also respect the invariant; otherwise, the outcome of the sampling process is not guaranteed to respect the invariant. This has implications not only for inference, but also for the choice of baseline during evaluation.

4.6 Encoding SRTs as a factor graph

In the previous section, we described how to build an SRT from a raw sentence. However, we need to find a way to label an SRT when it is presented to us unlabeled in held-out data in order to find the nodes that are involved in an opinionated expression. One approach would be to train local classifiers at each node to determine whether a node is part of an opinionated expression. However, the output of these local classifiers could be inconsistent. Rather than create a *post hoc* fix [Barutcuoglu et al., 2006], in this section, we develop machine learning tools to produce a labeled SRT in a single, unified model.

4.6.1 Factor graphs

A factor graph [Kschischang et al., 1998] is a representation of a joint probability distribution in the form of a graph with two types of vertices: variable vertices and factor vertices. Given a set of variables $Z = \{z_1 \dots z_n\}$, we connect them via factors $F = \{f_1 \dots f_m\}$. Factors are functions that represent relationships, i.e. probabilistic dependencies, among the variables; the product of all factors gives the complete joint distribution p. Each factor f_i can take as input some corresponding subset of variables Y_i from Z. We can then write the relationship as follows:

$$p(Z) \propto\equiv \prod_{k=1}^{m} f_k(Y_k)$$

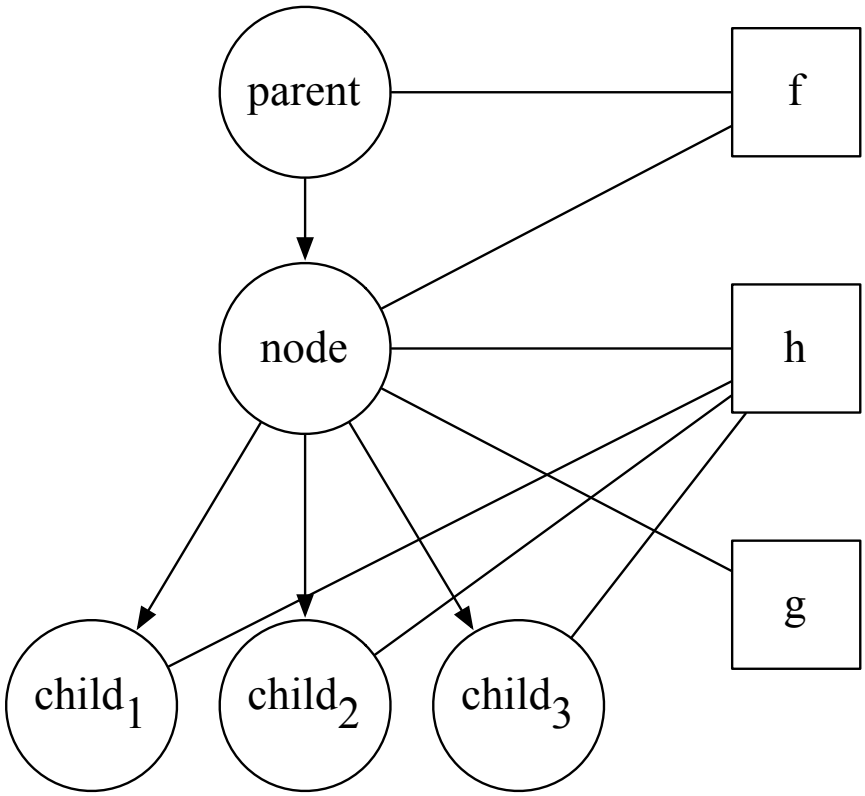
Our goal is to discover the values for the variables that best explain some set of data. While there are many approaches for inference in statistical models, we turn to MCMC methods [Neal, 1993] to discover the underlying structure of the model.

4.6.2 SRT model and sampler

A collection of syntactic relatedness tries has a correct labelling of every node with valid and invalid. The labels are our latent variables for which we are attempting to discover the posterior distribution. We employ the method of Finkel et al. [2005], to sample labelings.

The sampler requires an initial setting, one that respects the invariant. Our initial setting is produced by iterating through all labels in the SRT forest and randomly setting them as either valid or invalid with uniform probability. Any change in a setting, however, may change other labels as mentioned in the previous section in order to preserve the

Figure 4.3: Graphical model of SRT factors



invariant.

A Gibbs sampler samples new variable assignments from the conditional distribution, treating the variable assignments for all other variables fixed. However, the assignment of a single node is highly coupled with its neighbors, so a block sampler was used to propose changes to groups nodes that respect the overall validity of the overall assignments. This was implemented by changing the proposal distribution used by McCallum et al.'s FACTORIE framework [McCallum et al., 2009].

We can thus represent a node and its contribution to the overall score using the graph in Figure 4.3. This graph contains the given node, its parent, and a variable number of children. The factors that go into the labelling decision for each node are thus constrained

to a small, computationally tractable space around the given node. This graph contains three factors:

- g represents a function over features of the given node itself, or "node features."
- f represents a function over a bigram of features taken from the parent node and the given node, or "parent-node" features.
- h represents a function over a combination features on the node and features of *all* its children, or "node-child" features.

We provide further details about these factors in the next section.

We can represent the set of possible observed linguistic feature classes (part-of-speech, word, and so on) as the set Feat. Then we can represent Figure 4.3 in the form of a scoring function that reflects the contribution of each node to the score of a sample:

$$\begin{aligned}\text{score}(\text{label}|\text{node}) = \prod_{\phi \in \text{Feat}} & f(\text{parent}_\phi, \text{node}_\phi|\text{label}) \\ & g(\text{node}_\phi|\text{label}) \\ & h(\text{node}_\phi, \text{child}_{1\phi}, \ldots, \text{child}_{n\phi}|\text{label})\end{aligned}$$

After assignments for the latent variables are sampled, the weights for the factors (which when combined create individual factors f that define the joint) must be learned. This is accomplished via the *sample-rank* algorithm [Wick et al., 2009].

Table 4.1: Performance using various feature combinations, including some without enforcing the invariant. Mean averages and ranges for 10 runs.

Experiment	Features	Precision / Recall / F	Accuracy	Invariant?
Baseline	N/A	0.78 (0.73-0.86) / 0.06 (0.04-0.08) / 0.11 (0.08-0.15)	0.51 (0.51-0.53)	Yes
Baseline	N/A	0.50 (0.50-0.51) / 0.49 (0.49-0.50) / 0.50 (0.49-0.50)	0.50 (0.49-0.50)	No
Node only	All	0.63 (0.50-0.81) / 0.34 (0.21-0.49) / 0.42 (0.33-0.50)	0.54 (0.49-0.57)	Yes
Node only	All	0.51 (0.50-0.52) / 0.88 (0.82-0.92) / 0.65 (0.63-0.66)	0.51 (0.50-0.52)	No
Node only	All but word	0.63 (0.48-0.95) / 0.40 (0.01-0.73) / 0.42 (0.01-0.61)	0.53 (0.48-0.58)	Yes
Node only	All but word	0.57 (0.50-0.65) / 0.56 (0.30-0.84) / 0.55 (0.40-0.64)	0.55 (0.50-0.60)	No
Parent, node	Parent: all but word Node: all	0.71 (0.59-0.82) / 0.21 (0.11-0.28) / 0.31 (0.20-0.39)	0.55 (0.53-0.56)	Yes
Parent, node	All	0.84 (0.75-0.95) / 0.11 (0.03-0.19) / 0.19 (0.06-0.31)	0.53 (0.51-0.56)	Yes
Full graph	Parent: all but word Node: all but word Children: POS only	0.59 (0.50-0.67) / 0.39 (0.25-0.56) / 0.46 (0.36-0.54)	0.54 (0.49-0.60)	Yes
Full graph	Parent: all Node: all Children: all but word	0.67 (0.57-0.74) / 0.39 (0.25-0.54) / 0.47 (0.37-0.58)	0.59 (0.55-0.61)	Yes
Full graph	All	0.70 (0.62-0.79) / 0.35 (0.21-0.47) / 0.46 (0.32-0.55)	0.59 (0.56-0.62)	Yes
Full graph	All	0.67 (0.63-0.71) / 0.25 (0.19-0.30) / 0.36 (0.39-0.42)	0.56 (0.54-0.57)	No

4.7 Results and discussion

4.7.1 Evaluation measure

During the training phase, we evaluate the quality of a sample based on label accuracy. We also use label accuracy as an evaluation measure to compare the effects of

the factors and linguistic feature combinations on the performance of the system. We do not only need to retrieve the valid nodes, but we also avoid retrieving the invalid nodes in order to distinguish between relevant and irrelevant subcomponents.

Thus, we employ precision and recall as a performance metric. Given the invariant, recall is an indicator of how far down the correct valid paths we have marked. Higher recall means that the system has come closer to the correct valid leaves of the SRTs. Higher precision means that the system has successfully excluded the correct invalid labels.

We define recall and precision in terms of the validity label on each node. The valid label is the positive class. A true positive is counted whenever the model chooses a valid label for a node when it is valid in the test data's ground truth. A false positive occurs when an valid label is applied by the model when the ground truth is invalid. True and false negatives are defined in the analogous manner for invalid. Recall, precision, F_1, and accuracy emerge using the standard definitions as described in chapter 2.

An example of how this works can be seen by comparing figure 4.2(d) to figure 4.2(f), viewing the former as the gold standard and the latter as a hypothetical system output. If we run the evaluation over that single SRT and treat valid as the positive class, we find that 3 true positives, 1 false positive, 2 false negatives, and no true negatives. There are 6 labels in total. That yields 0.50 accuracy, 0.75 precision, 0.60 recall, and 0.67 F-measure.

We run every experiment 10 times and take the mean average and range of all measures. F-measure is calculated for each run and then averaged *post hoc*. We define an experiment as a full cycle of training a model and testing it on the held-out data.

Very high precision with low recall means that it is not finding very many valid nodes

by being overcautious about avoiding invalid nodes. But high recall with low or moderate precision is even worse: it becomes difficult to distinguish paths correctly followed with valid from paths incorrectly followed.

4.7.2 Experiments

Our baseline system is simply the initial setting of the labels for the sampler: uniformly random assignment of validity labels, respecting the invariant. This leads to a very large class imbalance in favour of invalid as any switch to invalid all descendent branches to convert to invalid, while any switch to valid causes only one branch of ancestors to convert to valid as well.

Our next systems involve combinations of our SRT factors with the observed linguistic features. All our experiments include the factor g that pertains only to the features of the node. Then we add factor f—the parent-node "bigram" features—and finally factor h, the variable-length node-child features. We also experiment be including and excluding combinations of POS, role, and word features.

We ran the Gibbs sampler for 200 iterations to train a model with a particular factor-feature combination. We use the learned model to predict the labels on the held-out testing data by running the inference algorithm (sampling labels only) for 50 iterations.

To examine the importance of the invariant enforced by the trie, we also explored models that only made local decisions. That is, we permitted the learner to acquire a model that does not find consistent paths through the SRTs but only finds valid nodes individually. We started with a baseline and initial setting that does not respect the invariant; instead,

each label is selected with uniform probability individually. The experiments had both the node-only features and the full feature set.

Since these models do not respect the invariant, local features alone determine the labelling. The interrupted paths make it difficult to use the output in techniques such as Nakagawa et al.'s polarity classifier, so they function as a baseline for the invariant-respecting models. This helps us determine whether the potentially radical changes in the trie labelling caused by enforcing the invariant hurts or helps the label accuracy.

4.7.2.1 Linguistic features

Aside from the validity label, each node contained the following linguistic features: the word from the sentence itself, the part-of-speech (POS) tag assigned by the Stanford parser, and the label of the incoming dependency edge. The POS tags come from the Penn Treebank tagset [Marcus et al., 1993] and include such things as "VBZ" (verb, 3rd person singular present) and "JJR" (comparative adjective).

Examples of dependency edge labels can be found in the graph in figure 4.1. We treat an incoming edge label as a feature of the node. Note that for these experiments, we treat the dependency graph as undirected.

4.7.3 Discussion

We present a sampling of possible feature-factor combinations in table 4.1 in order to show some of the overall trends in the performance of the system. First we analyze the invariant-respecting experiments, and then we discuss them in relation to the experiments

that ignore the invariant.

Unsurprisingly, the invariant-respecting baseline had very high precision but very low recall, due to the invariant. Simply including the node-only g factor with all features dramatically increases the recall, while hurting precision. We regain some recall without changing precision if we remove the word features. This suggests that some words in some SRTs are associated with valid labels in the training data, but not as much in the testing data.

If we include the parent-node f features along with all of the g features, we get an increase in precision and loss in recall. The parent-node word features appear, in this case, to help maintain high precision.

Including the node-child h features along with f and g has some interesting effects. If we include all the features on all factors, we preserve most of the precision but get a large increase in recall. Exclusion of the word h features increases the recall slightly more than it hurts the precision. When we exclude the word features for all factors and exclude the role h features, we hurt precision, recall, and accuracy.

The accuracy measure, however, does show overall improvement with the inclusion of more feature-factor combinations. In particular, the node-child h factor does appear to have an effect on the performance. The presence of certain combinations of child word, POS tags, and roles appear to provide some indication of the validity labelling of some of the nodes. The best models in terms of accuracy include all or almost all of the features.

One important detail is the ranges of these results, considering that we ran each experiment 10 times. Wider ranges indicate that the training and testing process is not very stable given the data; training does not converge on the similar models. What differs

between experiment runs are the random initial labellings. Using the node g factor without any other factor produces some very wide ranges particularly in recall. Using more factors and linguistic features seems to promote a certain amount of stability in performance and less sensitivity to the initial setting.

Our non-invariant-respecting random baseline unsurprisingly has nearly 50% precision, recall, and accuracy. Including the node-only features dramatically increases recall, less so if we exclude the word features. The word features appear to have a strong effect on recall just as in the invariant-respecting case with the node-only features.

We also ran the full factor graph with all our features without the invariant. The precision is dramatically improved, but with a large cost to recall. However, it also underperforms the equivalent invariant-respecting model in all measures, including accuracy. Nevertheless, it outperforms the node-only model in precision and accuracy. Manual inspection of the output of all these invariant-violating systems reveals rampant path discontinuity.

Though these invariant-violating models are unconstrained in the way they label the graph, our invariant-respecting models still outperform them. A coherent path contains more information than an incoherent one, considering that it is important to find negating and intensifying elements in context. Our SRT invariant allows us to achieve better performance while simultaneously extracting more utility.

The result ranges of individual runs in invariant-violating models tend to be smaller than those of the invariant-respecting models, particularly in the lower-performing simpler models. Experiments using all factors and features without the invariant have better performance than the simpler models, but their result ranges are larger, though not as large

as the invariant-respecting case.

4.7.4 Manual inspection

As there are two classes of node labels, there are two kinds of errors that can be made: replacement of valid labels with invalid labels, and vice versa. Each of these has different consequences. The classification of a valid as invalid means that an entire branch cannot be followed to a valid node containing an opinion word due to the invariant. The classification of an invalid label as valid means that a route is opened to classify some words as relevant opinion words when they are not.

One way of examining the kinds of errors made by the system is to look for patterns in misclassification of a node given its surrounding context, namely, its parent and children; this is the structure of the model. If any pattern is at all observable by human inspection, it will be in the relationships among parent and child POS tags, parent and child dependency edge labels, and parent and child words. However, even a reduced subset of tags will create a large number of combinations to examine, possibly too large to see anything but the most general patterns.

We inspected the output of the full graph model (with all linguistic features) on the held-out testing data. One pattern that prominently stood out was the misclassification of valid labels as invalid in the vicinity of Stanford dependency labels such as `conj_and`, the conjunction dependency; these kinds of labels have high "fertility" in that the labels that immediately follow in the given node's children could be a variety of types. In this case, this would create a data sparsity problem for learning.

This kind of data sparsity problem could be resolved by making certain kinds of features "transparent" to the learner. For example, if node q has an incoming dependency edge label of `conj_and`, then q's parent could also be directly connected to q's children, as a conjunction should be linguistically transparent to the status of the children in the sequence of validity labels.

There are many fewer incidents of invalid labels being classified as valid; this is not surprising, since, as with the initial random setting, the effect of making an invalid label valid has a smaller effect on other labels. Occasionally, there are paths through an SRT where an valid candidate word is the ancestor of an invalid candidate word from the set of candidates passed through crowdsourcing. The model sometimes appears to "overshoot" the valid candidate. Considering that recall is already fairly low, attempts to address this problem risks making the model too conservative. One potential solution is to prune or separate these paths that contain multiple crowdsourcing-labelled candidates.

4.7.4.1 Paths found

As to the value of the paths discovered, we examined the labelling on the held-out testing data of the best-performing model of the full graph system with all linguistic features. Here we look at some examples. Consider the following highlight group from the testing data:

> The contract is consistent with the desktop computing **Outsourcing** deals Citibank awarded EDS and Digital Equipment in 1996, and with the Citibank program announced last fall to improve customer service and efficiency, says

> Stan Welland, head of the bank's technology infrastructure unit.

Mechanical Turk workers identified "efficiency" as a relevant positive opinion about outsourcing. One of the paths discovered through the corresponding SRT by our system and correctly identified as valid is this one:

> Outsourcing:valid $\xrightarrow{\text{xcomp}}$ consistent:valid $\xrightarrow{\text{ccomp}}$ awarded:valid $\xrightarrow{\text{xcomp}}$ improve:valid $\xrightarrow{\text{dobj}}$ efficiency:valid

This and a number of similar paths through the SRT contain a string of generally positive words ending in "efficiency", adding evidence for the intensely positive context in which the word is being used about outsourcing. There are several non-local dependencies in this process, as evidenced by the complementizer dependencies ("xcomp", "ccomp").

However, the polarity of "efficiency" is already likely to be positive in this domain. Here is another example where the model identified an target-opinion word relation non-locally:

> But Microsoft's informal approach may not be enough as the number of **blogs** at the company grows, especially since the line between "personal" Weblogs and those done as part of the job can be hard to distinguish.

In this case, the workers decided that "distinguish" expressed a negative opinion about blogs, in the sense that something that was difficult to distinguish was a problem: the modifier "hard" is what makes it negative. The system found an entirely valid path that connected these attributes into a single unit:

> Blog:valid $\xrightarrow{\text{prep\_of}}$ number:valid $\xrightarrow{\text{nsubj}}$ grows:valid $\xrightarrow{\text{ccomp}}$ hard:valid $\xrightarrow{\text{xcomp}}$ distinguish:valid

In this path, "blog" and "distinguish" are both connected to one another, by "hard", giving "distinguish" its negative spin.

4.8 Conclusions and future work

In this work, we have applied recent advances in machine learning technology to produce the first steps in the robust modeling of syntactic structure for an information extraction application. A solution to the problem of modeling these structures requires the development of new techniques that model complex linguistic relationships in an application-dependent way. We have shown that we can mine these relationships without being overcome by the data-sparsity issues that typically stymie learning over complex linguistic structure.

Recent applications of syntactic features in opinion mining all require complete connectedness in syntactic structures. Both Jakob and Gurevych's approach and Nakagawa et al.'s approach rely on identifying connected subgraphs of dependency parses that contain a correct sequence of linguistic features. However, the former use syntactic features in a bag-of-features model that requires that opinionated language already be identified. The latter use a technique that learns patterns directly over grammatical structures, but they apply it to a binary classification technique. Syntactic connectedness is necessary to preserve sequences of both explicit and implicit negations as well as other features relevant to opinion.

The limitations on these techniques ultimately find their root in the difficulty in modeling complex syntactic structures that simultaneously exclude irrelevant portions of the structure while maintaining the connected relations. Our technique uses a structure-labelling scheme that enforces connectedness. It turns out that the enforcement of connected structure is not only necessary to produce useful results, it has a positive effect on

the accuracy of the structural labels.

Paths need not be complete to the end for use in applied contexts. In cases where lightweight techniques are used to identify a subset of higher-probability candidate opinion expressions, further evidence can be recovered from partial paths by, for example, using longer valid paths "aiming" at an opinion-word leaf as positive evidence for the relevance of that word. Needless to say, the longer we make the correct labeling of a path, the more ways in which this technique can be applied, but there remains ample opportunity for future work in this and related techniques, which we describe in the remainder of this section.

4.8.1 Feature engineering

We derived our linguistic features directly from the Stanford parser, and we only used the words themselves, the POS tags, and the dependency edge labels. We were able to show positive effects of our approach even with these simple features. Consequently, we expect to be able to do better by experimenting with better-derived features.

For example, the POS tagset used by the Stanford parser contains multiple verb tags that represent different English tenses and numbers. For the purpose of sentiment relations, it is possible that the differences between verb tags are too small to matter and are causing data sparsity issues. Thus, we could collapse some of these tag families into single tags. However, it could also turn out that there are significant differences in distribution of tenses and numbers in relation to opinion. Similar experimental questions exist for the dependency role features and for other grammatical representation schemes.

4.8.2 Objective function

In our machine learning approach, the quality of a sample was scored against a gold standard using accuracy as the measure. Every label is considered equal, no matter where it is in the trie. But our goal is to place correct, adjacent valid labels on paths as far down the trie as possible. Therefore, one possible experiment would be to use a quality measurement scheme that rewards correct valid labels at different rates—the further down the trie, the higher the reward.

Considering our emphasis on precision, however, another modification to the scoring function would be to punish incorrect valid labels at a greater rate than incorrect invalid labels. All of this would be built on top of a system whose effectiveness we have now demonstrated.

Chapter 5

Conclusions and future work

5.1 Summary

In chapter 1, we outlined a program of work in expanding and developing the techniques used for sentiment-related feature engineering and extraction, motivated by the gaps in existing techniques in this new area of research. Here we summarize some of our findings in the areas we addressed.

Chapter 2 discussed the development of a technique that identified paragraph-level vicinities of domain-specific opinion while ranking subjectivity features. We found that by using linguistically lightweight techniques from relation mining and vector-space modeling, we could obtain good performance in retrieving source-target relations. In the process, we found not only that features gleaned from a general-purpose subjectivity lexicon were effective in improving the model's performance, but also that we could use the technique to rank their prominance in the model in such a way that the most highly-ranked features were also domain-relevant.

Potential future work on this technique could include an additional effort to test and improve the scalability of this type of vector-space modeling of sentiment relations. Another possible direction would be to use the technique to bootstrap a larger within-domain sentiment lexicon using the top-ranked words it found in the general-purpose sentiment lexicon.

Chapter 3 developed a crowdsourcing technique to replicate a domain-specific metasubjective perspective in unskilled annotators, and used the result to create a resource for the work descripted in chapter 4. In chapter 3, we found that using trained annotators to identify both targets and and relevant words in large quantities of text was not an effective strategy in the context of the information technology (IT) business domain; it was difficult to reconcile the different notions of what an opinion really is between the annotators. Instead, our user interface design and quality control process eliminated some of the metasubjectivity from the annotation process, allowing us to obtain data that reasonably replicated the intentions of the task designers. Potential future work in this area includes testing alternate quality control and vote-aggregation schemes as well as trying a similar procedure on other linguistic problems.

Finally, chapter 4 described the first machine-learning framework for acquiring opinion word-target relations directly over the components of a parse tree. Using a suffix tree-based formalism, the syntactic relatedness trie (SRT), we were able to encode the grammatical relationships between target entities and relevant opinion words; this encoding was performed over paths through a dependency graph. Learning and inference were performed using Gibbs sampling over SRTs modeled as factor graphs.

We trained models using combinations of very elementary linguistic features with crowdsourced data from chapter 3 for training and testing, and we found that, overall, the use of more types of features led to higher performance and more stable results. Future work in this area can involve experimenting with more advanced feature combinations and with the objective function that guides the training process.

Chapters 3 and 4 formed a single procesing pipeline, with the work described in

the latter depending on the former. Chapter 2 described a lighter technique and was evaluated separately. In the following sections, we describe the manner in which they can be connected together, and how they ultimately form part of the same larger project.

5.2 Future work

Now that we have summarized the results of our experiments and provided some observations as to the future directions of our work, we discuss the potential for future work based on our contributions thus far. First, we discuss how these components might fit into the context of a larger end-to-end system. Then we describe the next component in the pipeline: polarity classification.

5.2.1 The pipeline

We have built and tested components that represent the beginning of an end-to-end pipeline starting from a large corpus of genre-specific articles to grammatical and lexical feature extraction, as well as paragraph-level opinion vicinity identification. We did not, however, connect these components together.

Our preselection of sentences was performed independently of the paragraph-level opinion detector using an even lighter technique. The rationale for this was to test the efficacy of crowdsourcing in this context alone without bias emerging from the output of the paragraph-level system, as this technique could be used in the context of other large-scope lightweight opinion detection systems.

However, our crowdsourcing technique was constructed primarily in the service of

our dependency parse node classification system in chapter 4, since no equivalent resource existed before then, even outside the information technology domain. Thus, these parts are already connected.

Connecting the paragraph-level opinion-mining system and the crowdsourcing technique, however, should be an immediate and obvious direction for future research. Furthermore, the paragraph-level opinion mining step not only helps to highlight opinionated mentions of target entity concepts, it also provides sentiment-related word features that could potentially be useful for improving the performance of the work in 4.

Finally, the component developed in chapter 4 terminates with a partial labelling of opinion-related dependency tree paths. Aside from the feature improvements already discussed at the end of that chapter, we also need to consider the downstream uses of the output of that system. We discuss this in the next section.

5.2.2 Polarity classification

5.2.2.1 Introduction

We now present a high-level sketch of a technique for opinion polarity classification, which depends on obtaining output of the system described in the previous chapter. We present a basic intuition about the use of the dependency paths to infer sentiment polarity. Then we describe a graphical model relating all the information we have collected using the techniques of the previous three chapters in such a manner that we can use Bayesian inference techniques to develop a polarity classifier for source-target pairs.

The main advantage of our approach is that, by this point, we have targeted the

specific language in a sentence that bears sentiment relevant to a given target. Previous work [Wilson, 2007] did not find opinion-bearing language with specific enough bounds that this could be used automatically in a polarity classifier.

5.2.2.2 Polarity and dependency

The system described in the previous chapter not only allows us to classify a (t, w) pair as reflecting the fact that the sentiment word w expresses an opinion about a target t, but it also produces one or more paths through a dependency graph that transitively connects t to w. Those paths contain items (words and grammatical roles) that we expect will allow us to classify the polarity of w with respect to t.

Consider the sentence

Bill refuses to behave well.

The (t, w) pair in question, "Bill" and "well", contain the following paths through the Stanford Parser's dependency tree for this sentence:

Bill $\xrightarrow{\text{nsubj}}$ refuses $\xrightarrow{\text{xcomp}}$ behave $\xrightarrow{\text{advmod}}$ well

Bill $\xrightarrow{\text{xsubj}}$ behave $\xrightarrow{\text{advmod}}$ well

If we were to take only the shorter path, we would assume that this refers to a positive opinion of Bill. It is on the longer path that we see that there is a word, "refuses", that negates the polarity of "well". With the use of further negative terms, we can extend this sentence into a positive description of Bill.

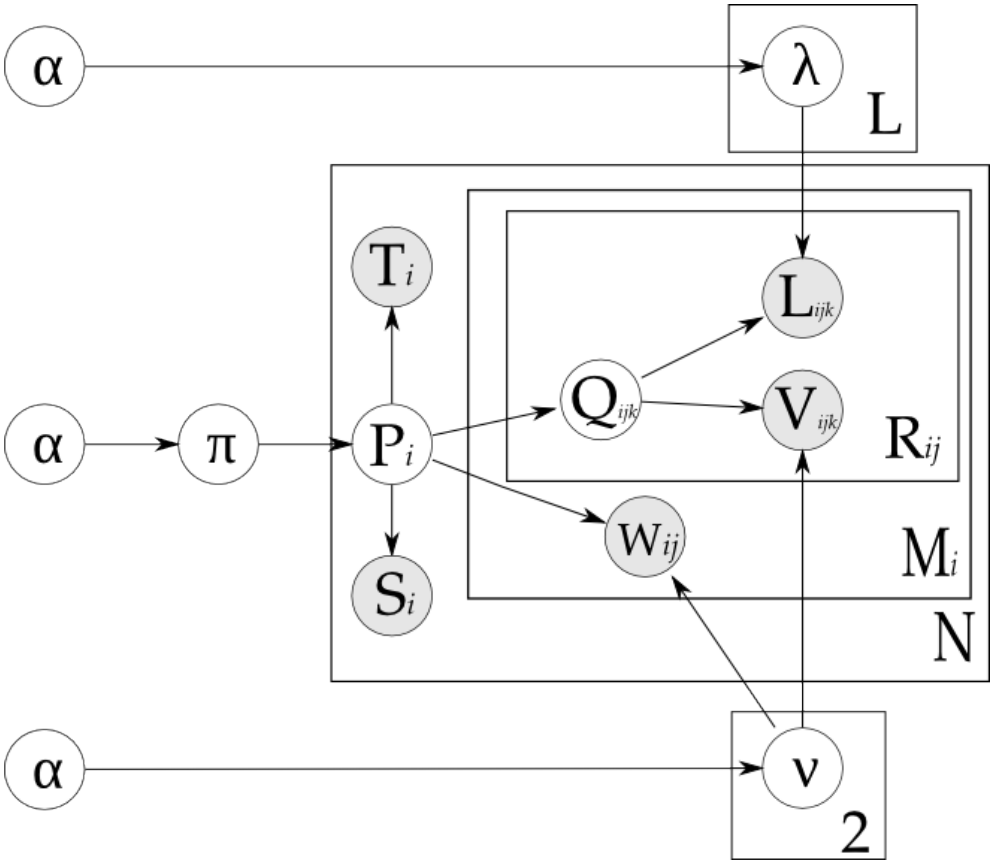


Figure 5.1: Graphical model of path-based polarity classifier.

Bill is reluctant to refuse to behave well.

The Stanford Parser now gives us only one path between "Bill" and "well".

Bill $\xrightarrow{\text{nsubj}}$ reluctant $\xrightarrow{\text{xcomp}}$ refuse $\xrightarrow{\text{xcomp}}$ behave $\xrightarrow{\text{advmod}}$ well

We can therefore conceive of sentiment as something that flows from w to t in (t, w) passing through each step of the path. Negating expressions act as gates which alter the polarity of the flow.

5.2.3 Graphical model

Ultimately, we are attempting to infer a single label—positive or negative polarity—from a complex but structured object that contains our given data: targets, related sentiment words, the paths between them, and perhaps sources. Thinking about this problem in a Bayesian style would lead us to describe our classification problem as the manner in which a polarity class biases the generation of this object.

This has led us to develop the model depicted in figure 5.1. We describe it as follows, first with the observed variables:

- For every $0..N$ opinions across the corpus held by a source S_i about a target T_i, $i < N$, there are M_i paths ending in sentiment words W_{ij}, $j < M_i$, that produce that opinion. In plain language, this simply reflects the fact that for every source-target pair, there may be multiple paths that reflect that source's opinion.

- For every $0..M_i$ paths that reflect an opinion, there are R_{ij} path elements leading to the sentiment word at the end. Each path element consists of a label L_{ijk} and a word V_{ijk}, $k < R_{ij}$

And here is how their generation is biased by the unobserved variables:

- Each path element (L_{ijk}, V_{ijk}) is the product of a polarity state Q_{ijk} for that point along the path.

- Label L_{ijk} is drawn from a set of label distributions λ that reflect the edge label categories from the Stanford parser.

- The word variable V_{ijk} is generated taking into account two classes distributions ν which reflect the negativity of the given word—whether it negates polarity or not.

- The path element polarity state variable Q_{ijk} is in turn influenced by the polarity of the source-target relationship itself, P_i.

- The the variable that represents the word at the end of the path, W_{ij}, is also influenced by P_i.

- P_i also controls the generation of possible source-target pairs, S_i and T_i. Between the three of them, we have the full $\{$source, target, opinion$\}$ triple.

- P_i is parameterized on π, representing the probability of either of the polarity classes.

- π, λ, and ν are generated by hyperparameters α.

The state space is therefore defined by π, the ν negativity variables (ν_{neg}, $\nu_{\text{non}-\text{neg}}$), the λ path edge label variables, a P_i polarity class for every source-target pair, and a Q_i polarity class for every path element. This unifies all the information we have recovered in the previous two chapters for polarity classification. Then we can compute Pr($P_i = \text{positive}$) and Pr($P_i = \text{negative}$) conditioned on all the other variables.

5.2.4 Plans for implementation

Describing polarity classification system enables us to use Monte Carlo Markov Chain methods such as Gibbs sampling [Resnik and Hardisty, 2009] to estimate the weights in the described above. Some of the work that we need to do is to derive the equations for

the sample and to choose the appropriate distributions for the parameters π, λ, and ν based on their associated hyperparameters.

The annotation effort described in the previous chapter is also the basis for the data generation step for the work in this chapter: assigning polarities to the targets and selecting the words on which those polarities are conditioned.

Bibliography

Cecilia Ovesdotter Alm. Subjective natural language problems: Motivations, applications, characterizations, and implications. In *ACL (Short Papers)*, 2011.

John Langshaw Austin. *How to do things with words*. Clarendon, Cambridge, Mass., 1962.

Zafer Barutcuoglu, Robert E. Schapire, and Olga G. Troyanskaya. Hierarchical multi-label prediction of gene function. *Bioinformatics*, 22(7):830–836, 2006.

Steven Bethard, Hong Yu, Ashley Thornton, Vasileios Hatzivassiloglou, and Dan Jurafsky. Automatic extraction of opinion propositions and their holders. In *Proceedings of the AAAI Spring Symposium on Exploring Attitude and Affect in Text*, 2004.

Daniel M. Bikel, Richard Schwartz, and Ralph M. Weischedel. An algorithm that learns what's in a name. *Mach. Learn.*, 34(1-3), 1999.

Johan Bollen, Huina Mao, and Xiao-Jun Zeng. Twitter mood predicts the stock market. *CoRR*, abs/1010.3003, 2010.

Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Identifying sources of opinions with conditional random fields and extraction patterns. In *HLT '05: Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2005.

Yejin Choi, Eric Breck, and Claire Cardie. Joint extraction of entities and relations for opinion recognition. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2006.

Marie-Catherine de Marneffe and Christopher D. Manning. The stanford typed dependencies representation. In *CrossParser '08: Coling 2008: Proceedings of the workshop on Cross-Framework and Cross-Domain Parser Evaluation*, Morristown, NJ, USA, 2008. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370, Stroudsburg, PA, USA, 2005. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1219840.1219885.

Roxana Girju, Preslav Nakov, Vivi Nastase, Stan Szpakowicz, Peter Turney, and Deniz Yuret. Semeval-2007 task 04: classification of semantic relations between nominals. In *SemEval '07: Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 13–18, Morristown, NJ, USA, 2007. Association for Computational Linguistics.

Aria Haghighi and Dan Klein. Simple coreference resolution with rich syntactic and semantic features. In *EMNLP '09: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, Morristown, NJ, USA, 2009. Association for Computational Linguistics.

Pei-Yun Hsueh, Prem Melville, and Vikas Sindhwani. Data quality from crowdsourcing: a study of annotation selection criteria. In *Proceedings of the NAACL HLT 2009 Workshop on Active Learning for Natural Language Processing*, HLT '09, Stroudsburg, PA, USA, 2009. Association for Computational Linguistics.

Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *EMNLP*, 2010.

T. Joachims. Making large-scale SVM learning practical. In B. Schölkopf, C. Burges, and A. Smola, editors, *Advances in Kernel Methods - Support Vector Learning*, chapter 11, pages 169–184. MIT Press, Cambridge, MA, 1999.

Jason S. Kessler, Miriam Eckert, Lyndsay Clark, and Nicolas Nicolov. The 2010 ICWSM JDPA sentment corpus for the automotive domain. In *4th Int'l AAAI Conference on Weblogs and Social Media Data Workshop Challenge (ICWSM-DWC 2010)*, 2010.

S-M. Kim and Eduard Hovy. Identifying opinion holders for question answering in opinion texts. Proceedings of AAAI-05 workshop on Question Answering in Restricted Domains. Pittsburgh, Pennsylvania, 2005.

Soo-Min Kim and Eduard Hovy. Extracting opinions, opinion holders, and topics expressed in online news media text. In *SST '06: Proceedings of the Workshop on Sentiment and Subjectivity in Text*, pages 1–8, Morristown, NJ, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-75-2.

Moshe Koppel and Jonathan Schler. The importance of neutral examples for learning sentiment. *Computational Intelligence*, 22(2), 2006.

Frank R. Kschischang, Brendan J. Frey, and Hans andrea Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on Information Theory*, 47:498–519, 1998.

Sandra Kübler, Ryan McDonald, and Joakim Nivre. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 2(1), 2009.

Mitchell P. Marcus, Beatrice Santorini, and Mary A. Marcinkiewicz. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2): 313–330, 1993.

Andrew McCallum, Karl Schultz, and Sameer Singh. Factorie: Probabilistic programming via imperatively defined factor graphs. In *Neural Information Processing Systems (NIPS)*, 2009.

Donald Metzler and W. Bruce Croft. Combining the language model and inference network approaches to retrieval. *Information Processing and Management*, 40(5):735 – 750, 2004.

Karo Moilanen and Stephen Pulman. Sentiment composition. In *Proceedings of the Recent Advances in Natural Language Processing International Conference (RANLP-2007)*, Borovets, Bulgaria, 2007.

Raymond J. Mooney and Razvan Bunescu. Mining knowledge from text using information extraction. *SIGKDD Explor. Newsl.*, 7(1):3–10, 2005.

José E. Moreira, Maged M. Michael, Dilma Da Silva, Doron Shiloach, Parijat Dube, and Li Zhang. Scalability of the nutch search engine. In Burton J. Smith, editor, *ICS*, pages 3–12. ACM, 2007.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *HLT-NAACL*, 2010.

Radford M. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, University of Toronto, 1993.

Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Found. Trends Inf. Retr.*, 2(1-2), 2008.

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment classification using machine learning techniques. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 79–86, 2002.

Philip Resnik and Eric Hardisty. Gibbs sampling for the uninitiated, 2009. http://www.umiacs.umd.edu/~resnik/pubs/gibbs.pdf.

Josef Ruppenhofer, Swapna Somasundaran, and Janyce Wiebe. Finding the sources and targets of subjective expressions. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odjik, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, 2008. European Language Resources Association (ELRA).

Asad B. Sayeed and Amy Weinberg. Complementizer-stranding by phase. `http://ling.auf.net/lingBuzz/001231`, 2009.

Asad B. Sayeed, Timothy J. Meyer, Hieu C. Nguyen, Olivia Buzek, and Amy Weinberg. Crowdsourcing the evaluation of a domain-adapted named entity recognition system. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics, 2010a.

Asad B. Sayeed, Hieu C. Nguyen, Timothy J. Meyer, and Amy Weinberg. Expresses-an-opinion-about: using corpus statistics in an information extraction approach to opinion mining. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, 2010b.

Asad B. Sayeed, Bryan Rusk, Martin Petrov, Hieu C. Nguyen, Timothy J. Meyer, and Amy Weinberg. Crowdsourcing syntactic relatedness judgements for opinion mining in the study of information technology adoption. In *Proceedings of the Association for Computational Linguistics 2011 workshop on Language Technology for Cultural Heritage, Social Sciences, and the Humanities (LaTeCH)*. Association for Computational Linguistics, 2011.

Rion Snow, Brendan O'Connor, Daniel Jurafsky, and Andrew Y. Ng. Cheap and fast—but is it good?: evaluating non-expert annotations for natural language tasks. In *EMNLP 2008*, 2008.

Swapna Somasundaran and Janyce Wiebe. Recognizing stances in online debates. 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: Volume 1*, ACL '09, 2009.

Philip J. Stone and Earl B. Hunt. A computer approach to content analysis: studies using the general inquirer system. In *AFIPS '63 (Spring): Proceedings of the May 21-23, 1963, spring joint computer conference*, New York, NY, USA, 1963. ACM.

Veselin Stoyanov and Claire Cardie. Partially supervised coreference resolution for opinion summarization through structured rule learning. In *EMNLP '06: Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 336–344, Morristown, NJ, USA, 2006. Association for Computational Linguistics. ISBN 1-932432-73-6.

Songbo Tan, Gaowei Wu, Huifeng Tang, and Xueqi Cheng. A novel scheme for domain-transfer problem in the context of sentiment analysis. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, CIKM '07, New York, NY, USA, 2007.

Peter D. Turney and Michael L. Littman. Measuring praise and criticism: Inference of semantic orientation from association. *ACM Transactions on Information Systems (TOIS)*, 21(4):315–346, 2003.

Michael Wick, Khashayar Rohanimanesh, Aron Culotta, and Andrew Mccallum. SampleRank: Learning preference from atomic gradients. In *NIPS WS on Advances in Ranking*, 2009.

Theresa Wilson. Annotating opinions in the world press. In *In SIGdial-03*, 2003.

Theresa Wilson. *Fine-grained Subjectivity and Sentiment Analysis: Recognizing the Intensity, Polarity, and Attitudes of private states*. PhD thesis, Intelligent Systems Program, University of Pittsburgh, 2007.

Theresa Wilson and Janyce Wiebe. Annotating attributions and private states. In *CorpusAnno '05: Proceedings of the Workshop on Frontiers in Corpus Annotations II*, Morristown, NJ, USA, 2005. Association for Computational Linguistics.

Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. OpinionFinder: A system for subjectivity analysis. In *HLT/EMNLP*, 2005a.

Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *HLT/EMNLP*, 2005b.

Li Zhuang, Feng Jing, and Xiaoyan Zhu. Movie review mining and summarization. In *CIKM*, 2006.