A Statistical Parsing Framework for Sentiment Classification

Li Dong[*], [**]
Beihang University

Furu Wei[†], [‡]
Microsoft Research

Shujie Liu[†]
Microsoft Research

Ming Zhou[†]
Microsoft Research

Ke Xu[*]
Beihang University

We present a statistical parsing framework for sentence-level sentiment classification in this article. Unlike previous works that use syntactic parsing results for sentiment analysis, we develop a statistical parser to directly analyze the sentiment structure of a sentence. We show that complicated phenomena in sentiment analysis (e.g., negation, intensification, and contrast) can be handled the same way as simple and straightforward sentiment expressions in a unified and probabilistic way. We formulate the sentiment grammar upon Context-Free Grammars (CFGs), and provide a formal description of the sentiment parsing framework. We develop the parsing model to obtain possible sentiment parse trees for a sentence, from which the polarity model is proposed to derive the sentiment strength and polarity, and the ranking model is dedicated to selecting the best sentiment tree. We train the parser directly from examples of sentences annotated only with sentiment polarity labels but without any syntactic annotations or polarity annotations of constituents within sentences. Therefore we can obtain training data easily. In particular, we train a sentiment parser, s.parser, from a large amount of review sentences with users' ratings as rough sentiment polarity labels. Extensive experiments on existing benchmark data sets show significant improvements over baseline sentiment classification approaches.

[*] State Key Laboratory of Software Development Environment, Beihang University, XueYuan Road No.37, HaiDian District, Beijing, P.R. China 100191. E-mail: `donglixp@gmail.com`; `kexu@nlsde.buaa.edu.cn`.

[**] Contribution during internship at Microsoft Research.

[†] Natural Language Computing Group, Microsoft Research Asia, Building 2, No. 5 Danling Street, Haidian District, Beijing, P.R. China 100080. E-mail: {`fuwei`, `shujliu`, `mingzhou`}@microsoft.com.

[‡] Corresponding author.

Submission received: 10 December 2013; revised version received: 26 July 2014; accepted for publication: 28 January 2015.

doi:10.1162/COLI_a_00221

1. Introduction

Sentiment analysis (Pang and Lee 2008; Liu 2012) has received much attention from both research and industry communities in recent years. Sentiment classification, which identifies sentiment polarity (positive or negative) from text (sentence or document), has been the most extensively studied task in sentiment analysis. Until now, there have been two mainstream approaches for sentiment classification. The lexicon-based approach (Turney 2002; Taboada et al. 2011) aims to aggregate the sentiment polarity of a sentence from the polarity of words or phrases found in the sentence, and the learning-based approach (Pang, Lee, and Vaithyanathan 2002) treats sentiment polarity identification as a special text classification task and focuses on building classifiers from a set of sentences (or documents) annotated with their corresponding sentiment polarity.

The lexicon-based sentiment classification approach is simple and interpretable, but suffers from scalability and is inevitably limited by sentiment lexicons that are commonly created manually by experts. It has been widely recognized that sentiment expressions are colloquial and evolve over time very frequently. Taking tweets from Twitter[1] and movie reviews on IMDb[2] as examples, people use very casual language as well as informal and new vocabulary to comment on general topics and movies. In practice, it is not feasible to create and maintain sentiment lexicons to capture sentiment expressions with high coverage. On the other hand, the learning-based approach relies on large annotated samples to overcome the vocabulary coverage and deals with variations of words in sentences. Human ratings in reviews (Maas et al. 2011) and emoticons in tweets (Davidov, Tsur, and Rappoport 2010; Zhao et al. 2012) are extensively used to collect a large number of training corpora to train the sentiment classifier. However, it is usually not easy to design effective features to build the classifier. Among others, unigrams have been reported as the most effective features (Pang, Lee, and Vaithyanathan 2002) in sentiment classification.

Handling complicated expressions delivering people's opinions is one of the most challenging problems in sentiment analysis. Compositionalities such as negation, intensification, contrast, and their combinations are typical cases. We show some concrete examples here:

(1) The movie is not *good*. [negation]

(2) The movie is very *good*. [intensification]

(3) The movie is not *funny* at all. [negation + intensification]

(4) The movie is *just so so*, but i still *like* it. [contrast]

(5) The movie is not very *good*, but i still *like* it. [negation + intensification + contrast]

The negation expressions, intensification modifiers, and the contrastive conjunction can change the polarity (Examples (1), (3), (4), (5)), strength (Examples (2), (3), (5)), or both (Examples (3), (5)) of the sentiment of the sentences. We do not need any detailed explanations here as they can be commonly found and easily understood in people's

[1] http://twitter.com.

[2] http://www.imdb.com.

daily lives. Existing works to address these issues usually rely on syntactic parsing results either used as features (Choi and Cardie 2008; Moilanen, Pulman, and Zhang 2010) in learning-based methods or hand-crafted rules (Moilanen and Pulman 2007; Jia, Yu, and Meng 2009; Klenner, Petrakis, and Fahrni 2009; Liu and Seneff 2009) in lexicon-based methods. However, even with the difficulty and feasibility of deriving the sentiment structure from syntactic parsing results put aside, it is an even more challenging task to generate stable and reliable parsing results for text that is ungrammatical in nature and has a high ratio of out-of-vocabulary words. The accuracy of the linguistic parsers trained on standard data sets (e.g., the Penn Treebank [Marcus, Marcinkiewicz, and Santorini 1993]) drops dramatically on user-generated-content (reviews, tweets, etc.), which is actually the prime focus of sentiment analysis algorithms. The error, unfortunately, will propagate downstream in the process of sentiment analysis methods building upon parsing results.

We therefore propose directly analyzing the sentiment structure of a sentence. The nested structure of sentiment expressions can be naturally modeled in a similar fashion as statistical syntactic parsing, which aims to find the linguistic structure of a sentence. This idea creates many opportunities for developing sentiment classifiers from a new perspective. The most challenging problem and barrier in building a statistical sentiment parser lies in the acquisition of training data. Ideally, we need examples of sentences annotated with polarity for the whole sentence as well as sentiment tags for constituents within a sentence, as with the Penn TreeBank for training traditional linguistic parsers. However, this is not practical as the annotations will be inevitably time-consuming and require laborious human efforts. Therefore, it is better to learn the sentiment parser only utilizing examples annotated with the polarity label of the whole sentence. For example, we can collect a huge number of publicly available reviews and rating scores on the Web. People may use *the movie is gud* ("gud" is a popular informal expression of "good") to express a positive opinion towards a movie, and *not a fan* to express a negative opinion. Also, we can find review sentences such as *The movie is gud, but I am still not a fan* to indicate a negative opinion. We can then use these two fragments and the overall negative opinion of the sentence to deduce sentiment rules automatically from data. These sentiment fragments and rules can be used to analyze the sentiment structure for new sentences.

In this article, we propose a statistical parsing framework to directly analyze the structure of a sentence from the perspective of sentiment analysis. Specifically, we formulate a Context-Free Grammar (CFG)–based sentiment grammar. We then develop a statistical parser to derive the sentiment structure of a sentence. We leverage the CYK algorithm (Cocke 1969; Younger 1967; Kasami 1965) to conduct bottom–up parsing, and use dynamic programming to accelerate computation. Meanwhile, we propose using the polarity model to derive sentiment strength and polarity of a sentiment parse tree, and the ranking model to select the best one from the sentiment parsing results. We train the parser directly from examples of sentences annotated with sentiment polarity labels instead of syntactic annotations and polarity annotations of constituents within sentences. Therefore we can obtain training data easily. In particular, we train a sentiment parser, named **s.parser**, from a large number of review sentences with users' ratings as rough sentiment polarity labels. The statistical parsing–based approach builds a principled and scalable framework to support the sentiment composition and inference which cannot be well handled by bag-of-words approaches. We show that complicated phenomena in sentiment analysis (e.g., negation, intensification, and contrast) can be handled the same way as simple and straightforward sentiment expressions in a unified and probabilistic way.

The major contributions of the work presented in this article are as follows.

- We propose a statistical parsing framework for sentiment analysis that is capable of analyzing the sentiment structure for a sentence. This framework can naturally handle compositionality in a probabilistic way. It can be trained from sentences annotated with only sentiment polarity but without any syntactic annotations or polarity annotations of constituents within sentences.
- We present the parsing model, polarity model, and ranking model in the proposed framework, which are formulated and can be improved independently. It provides a principled and flexible approach to sentiment classification.
- We implement the statistical sentiment parsing framework, and conduct experiments on several benchmark data sets. The experimental results show that the proposed framework and algorithm can significantly outperform baseline methods.

The remainder of this article is organized as follows. We introduce related work in Section 2. We present the statistical sentiment parsing framework, including the parsing model, polarity model, and ranking model, in Section 3. Learning methods for our model are explained in Section 4. Experimental results are reported in Section 5. We conclude this article with future work in Section 6.

2. Related Work

In this section, we give a brief introduction to related work about sentiment classification (Section 2.1) and parsing (Section 2.2). We tackle the sentiment classification problem in a parsing manner, which is a significant departure from most previous research.

2.1 Sentiment Classification

Sentiment classification has been extensively studied in the past few years. In terms of text granularity, existing works can be divided into phrase-level, sentence-level, or document-level sentiment classification. We focus on sentence-level sentiment classification in this article. Regardless of what granularity the task is performed on, existing approaches deriving sentiment polarity from text fall into two major categories, namely, lexicon-based and learning-based approaches.

The lexicon-based sentiment analysis uses dictionary matching on a predefined sentiment lexicon to derive sentiment polarity. These methods often use a set of manually defined rules to deal with the negation of polarity. Turney (2002) proposed using the average sentiment orientation of phrases, which contains adjectives or adverbs, in a review to predict its sentiment orientation. Yu and Hatzivassiloglou (2003) calculated a modified log-likelihood ratio for every word by the co-occurrences with positive and negative seed words. To determine the polarity of a sentence, they compare the average log-likelihood value with threshold. Taboada et al. (2011) presented a lexicon-based approach for extracting sentiment from text. They used dictionaries of words with annotated sentiment orientation (polarity and strength) while incorporating intensification and negation. The lexicon-based methods often achieve high precisions and do not need any labeled samples. But they suffer from coverage and domain adaption problems. Moreover, lexicons are often built and used without considering the context (Wilson, Wiebe, and Hoffmann 2009). Also, hand-crafted rules are often matched heuristically.

The sentiment dictionaries used for lexicon-based sentiment analysis can be created manually, or automatically using seed words to expand the list of words. Kamps et al. (2004) and Williams and Anand (2009) used various lexical relations (such as synonym and antonym relations) in WordNet to expend a set of seed words. Some other methods learn lexicons from data directly. Hatzivassiloglou and McKeown (1997) used a log-linear regression model with conjunction constraints to predict whether conjoined adjectives have similar or different polarities. Combining conjunction constraints across many adjectives, a clustering algorithm separated the adjectives into groups of different polarity. Finally, adjectives were labeled as positive or negative. Velikovich et al. (2010) constructed a term similarity graph using the cosine similarity of context vectors. They performed graph propagation from seeds on the graph, obtaining polarity words and phrases. Takamura, Inui, and Okumura (2005) regarded the polarity of words as spins of electrons, using the mean field approximation to compute the approximate probability function of the system instead of the intractable actual probability function. Kanayama and Nasukawa (2006) used tendencies for similar polarities to appear successively in contexts. They defined density and precision of coherency to filter neutral phrases and uncertain candidates. Choi and Cardie (2009a) and Lu et al. (2011) transformed the lexicon learning to an optimization problem, and used integer linear programming to solve it. Kaji and Kitsuregawa (2007) defined the χ^2-based polarity value and PMI-based polarity value as a polarity strength to filter neutral phrases. de Marneffe, Manning, and Potts (2010) utilized review data to define polarity strength as the expected rating value. Mudinas, Zhang, and Levene (2012) used word count as a feature template and trained a classifier using Support Vector Machines with linear kernel. They then regarded the weights as polarity strengths. Krestel and Siersdorfer (2013) generated topic-dependent lexicons from review articles by incorporating topic and rating probabilities and defined the polarity strength based on the results. In this article, the lexical relations defined in WordNet are not used because of its coverage. Furthermore, most of these methods define different criteria to propagate polarity information of seeds, or use optimization algorithms and sentence-level sentiment labels to learn polarity strength values. Their goal is to balance the precision and recall of learned lexicons. We also learn the polarity strength values of phrases from data. However, our primary objective is to obtain correct sentence-level polarity labels, and use them to form the sentiment grammar.

Learning-based sentiment analysis uses machine learning methods to classify sentences or documents into two (negative and positive) or three (negative, positive, and neutral) classes. Previous research has shown that sentiment classification is more difficult than traditional topic-based text classification, despite the fact that the number of classes in sentiment classification is smaller than that in topic-based text classification (Pang and Lee 2008). Pang, Lee, and Vaithyanathan (2002) investigated three machine learning methods to produce automated classifiers to generate class labels for movie reviews. They tested them on Naïve Bayes, Maximum Entropy, and Support Vector Machine (SVM), and evaluated the contribution of different features including unigrams, bigrams, adjectives, and part-of-speech tags. Their experimental results suggested that a SVM classifier with unigram presence features outperforms other competitors. Pang and Lee (2004) separated subjective portions from the objective by finding minimum cuts in graphs to achieve better sentiment classification performance. Matsumoto, Takamura, and Okumura (2005) used text mining techniques to extract frequent subsequences and dependency subtrees, and used them as features of SVM. McDonald et al. (2007) investigated a global structured model for jointly classifying polarity at different levels of granularity. This model allowed classification decisions from one level in the text to influence decisions at another. Yessenalina, Yue, and Cardie (2010) used sentence-level latent variables to improve document-level prediction. Täckström and McDonald (2011a) presented a latent variable model for only using document-level annotations to learn sentence-level sentiment labels, and Täckström and McDonald (2011b) improved it by using a semi-supervised latent variable model to utilize manually crafted sentence labels. Agarwal et al. (2011) and Tu et al. (2012) explored part-of-speech tag features and tree-kernel. Wang and Manning (2012) used SVM built over Naïve Bayes log-count ratios as feature values to classify polarity. They showed that SVM was better at full-length reviews, and Multinomial Naïve Bayes was better at short-length reviews. Liu, Agam, and Grossman (2012) proposed a set of heuristic rules based on dependency structure to detect negations and sentiment-bearing expressions. Most of these methods are built on bag-of-words features, and sentiment compositions are handled by manually crafted rules. In contrast to these models, we derive polarity labels from tree structures parsed by the sentiment grammar.

There have been several attempts to assume that the problem of sentiment analysis is compositional. Sentiment classification can be solved by deriving the sentiment of a complex constituent (sentence) from the sentiment of small units (words and phrases) (Moilanen and Pulman 2007; Klenner, Petrakis, and Fahrni 2009; Choi and Cardie 2010; Nakagawa, Inui, and Kurohashi 2010). Moilanen and Pulman (2007) proposed using delicate written linguistic patterns as heuristic decision rules when computing the sentiment from individual words to phrases and finally to the sentence. The manually compiled rules were powerful enough to discriminate between the different sentiments in *effective remedies* (positive) / *effective torture* (negative), and in *too colorful* (negative) and *too sad* (negative). Nakagawa, Inui, and Kurohashi (2010) leveraged a conditional random field model to calculate the sentiment of all the parsed elements in the dependency tree and then generated the overall sentiment. It had an advantage over the rule-based approach (Moilanen and Pulman 2007) in that it did not explicitly denote any sentiment designation to words or phrases in parse trees. Instead, it modeled their sentiment polarity as latent variables with a certain probability of being positive or negative. Councill, McDonald, and Velikovich (2010) used a conditional random field model informed by a dependency parser to detect the scope of negation for sentiment analysis. Some other methods model sentiment compositionality in the vector space. They regard the composition operator as a matrix, and use matrix-vector multiplication to obtain the transformed vector representation. Socher et al. (2012) proposed a recursive neural network model that learned compositional vector representations for phrases and sentences. Their model assigned a vector and a matrix to every node in a parse tree. The vector captured the inherent meaning of the constituent, and the matrix captured how it changes the meaning of neighboring words or phrases. Socher et al. (2013) recently introduced a sentiment treebank based on the results of the Stanford parser (Klein and Manning 2003). The sentiment treebank included polarity labels of phrases that are annotated using Amazon Mechanical Turk. The authors trained recursive neural tensor networks on the sentiment treebank. For a new sentence, the model predicted polarity labels based on the syntactic parse tree, and used tensors to handle compositionality in the vector space. Dong et al. (2014) proposed utilizing multiple composition functions in recursive neural models and learning to select them adaptively. Most previous methods are either rigid in terms of handcrafted rules, or sensitive to the performance of existing syntactic parsers they use. This article addresses sentiment compositions by defining sentiment grammar and borrowing some techniques in the parsing research field. Moreover, our method uses symbolic representations instead of vector spaces.

2.2 Syntactic Parsing and Semantic Parsing

The work presented in this article is close to traditional statistical parsing, as we borrow some algorithms to build the sentiment parser. Syntactic parsers are learned from the Treebank corpora, and find the most likely parse tree with the largest probability. In this article, we borrow some well-known techniques from syntactic parsing methods (Charniak 1997; Charniak and Johnson 2005; McDonald, Crammer, and Pereira 2005; Kübler, McDonald, and Nivre 2009), such as the CYK algorithm and Context-Free Grammar. These techniques are used to build the sentiment grammar and parsing model. They provide a natural way of defining the structure of sentiment trees and parse sentences to trees. The key difference lies in that our task is to calculate the polarity label of a sentence, instead of obtaining the parse tree. We only have sentence-polarity pairs as our training instances instead of annotated tree structures. Moreover, in the decoding process, our goal is to compute correct polarity labels by representing sentences as latent sentiment trees. Recently, Hall, Durrett, and Klein (2014) developed a discriminative constituency parser using rich surface features, adapting it to sentiment analysis. Besides extracting unigrams and bigrams as features, they learned interactions between tags and words located at the beginning or the end of spans. However, their method relies on phrase-level polarity annotations.

Semantic parsing is another body of work related to this article. A semantic parser is used to parse meaning representations for given sentences. Most existing semantic parsing works (Zelle and Mooney 1996; Kate and Mooney 2006; Raymond and Mooney 2006; Zettlemoyer and Collins 2007, 2009; Li, Liu, and Sun 2013) relied on fine-grained annotations of target logical forms, which required the supervision of experts and are relatively expensive. To balance the performance and the amount of human annotation, some works used only question-answer pairs or even binary correct/incorrect signals as their input. Clarke et al. (2010) used a binary correct/incorrect signal of a database query to map sentences to logical forms. It worked with FunQL language and transformed semantic parsing as an integer linear programming (ILP) problem. In each iteration, it solved ILP and updated the parameters of structural SVM. Liang, Jordan, and Klein (2013) learned a semantic parser from question-answer pairs, where the logical form was modeled as a latent tree-based semantic representation. Krishnamurthy and Mitchell (2012) presented a method for training a semantic parser using a knowledge base and an unlabeled text corpus, without any individually annotated sentences. Artzi and Zettlemoyer (2013) used various types of weak supervision to learn a grounded Combinatory Categorial Grammar semantic parser, which took context into consideration. Bao et al. (2014) presented a translation-based weakly supervised semantic parsing method to translate questions to answers based on CYK parsing. A log-linear model is defined to score derivations. All these weakly supervised semantic parsing methods learned to transform a natural language sentence to its semantic representation without annotated logical form. In this work, we build a sentiment parser. Specifically, we use a modified version of the CYK algorithm that parses sentences in a bottom–up fashion. We use the log-linear model to score candidates generated by beam search. Instead of using question-answer pairs, sentence-polarity pairs are used as our weak supervisions. We also use the parameter estimation algorithm proposed by Liang, Jordan, and Klein (2013).

3. Statistical Sentiment Parsing

We present the statistical parsing framework for sentence-level sentiment classification in this section. The underlying idea is to model sentiment classification as a statistical parsing process. Figure 1 shows the overview of the statistical sentiment parsing framework. There are three major components. The input sentence s is transformed into and represented by sentiment trees derived from the parsing model (Section 3.2), using the sentiment grammar defined in Section 3.1. Trees are scored by the ranking model in Section 3.3. The sentiment tree with the highest ranking score is treated as the best derivation for s. Furthermore, the polarity model (Section 3.4) is used to compute polarity values for the sentiment trees.

Notably, the sentiment trees t are unobserved during training. We can only observe the sentence s and its polarity label y in training data. In other words, we train the model directly from the examples of sentences annotated only with sentiment polarity labels but without any syntactic annotations or polarity annotations of the constituents within sentences. To be specific, we first learn the sentiment grammar and the polarity model from data as described in Section 4.2. Then, given the sentence and polarity label pairs (s, y), we search the latent sentiment trees t and estimate the parameters of the ranking model as detailed in Section 4.1.

To better illustrate the whole process, we describe the sentiment parsing procedure using an example sentence, *The movie is not very good, but i still like it*. The sentiment polarity label of the above sentence is "positive." There is negation, intensification, and contrast in this example, which are difficult to capture using bag-of-words classification methods. This sentence is a complex case that demonstrates the capability of the proposed statistical sentiment parsing framework, which motivates the work in this article. The statistical sentiment parsing algorithm may generate a number of sentiment trees for the input sentence. Figure 2 shows the best sentiment parse tree. It shows that the statistical sentiment parsing framework can deal with the compositionality of sentiment in a natural way. In Table 1, we list the sentiment rules used during the parsing process. We show the generation process of the sentiment parse tree from the bottom–up and the calculation of sentiment strength and polarity for every text span in the parsing process.

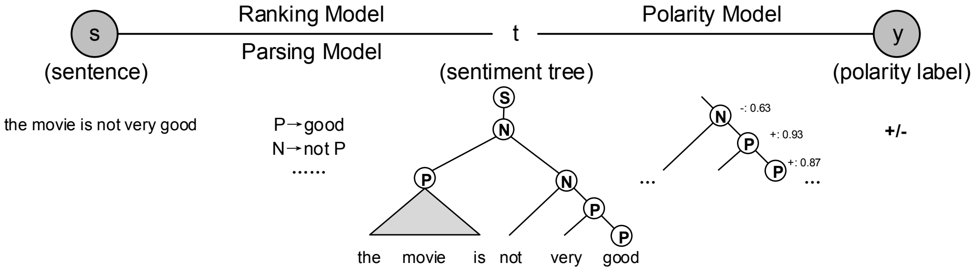


Figure 1
The parsing model and ranking model are used to transform the input sentence s to the sentiment tree t with the highest ranking score. Moreover, the polarity model defines how to compute polarity values for the rules of the sentiment grammar. The sentiment tree t is evaluated with respect to the polarity model to produce the polarity label y.

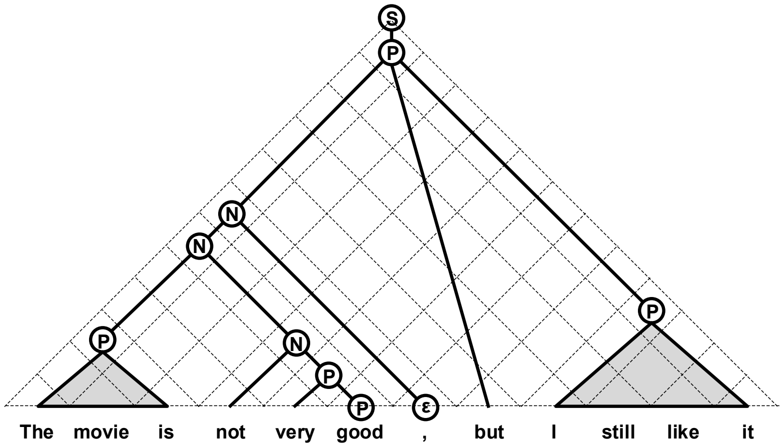


Figure 2
Sentiment structure for the sentence *The movie is not very good, but i still like it*. The rules used in the derivation process include $\{P \rightarrow$ the movie is; $P \rightarrow$ good; $P \rightarrow$ i still like it; $P \rightarrow$ very P; $N \rightarrow$ not P; $N \rightarrow PN$; $N \rightarrow N\mathcal{E}$; $\mathcal{E} \rightarrow$,; $P \rightarrow N$ but P; $S \rightarrow P\}$.

In the following sections, we first provide a formal description of the sentiment grammar in Section 3.1. We then present the details of the parsing model in Section 3.2, the ranking model in Section 3.3, and the polarity model in Section 3.4.

3.1 Sentiment Grammar

We develop the sentiment grammar upon CFG (Context-Free Grammar) (Chomsky 1956). Let $\mathcal{G} =< V, \Sigma, S, R >$ denote a CFG, where V is a finite set of non-terminals, Σ is a finite set of terminals (disjointed from V), $S \in V$ is the start symbol, and R is a set of rewrite rules (or production rules) of the form $A \rightarrow c$ where $A \in V$ and $c \in (V \cup \Sigma)^*$. We use $\mathcal{G}_s =< V_s, \Sigma_s, S, R_s >$ to denote the sentiment grammar in this article.

Table 1
Parsing process for the sentence *The movie is not very good, but i still like it*. $[i, Y, j]$ represents the text spanning from i to j is derived to symbol Y. N and P are non-terminals in the sentiment grammar, and $\mathcal{N}$ and $\mathcal{P}$ represent polarities of sentiment.

Span	Rule	Strength	Polarity
$[0, P, 3]$: the movie is	$P \rightarrow$ the movie is	0.52	$\mathcal{P}$
$[5, P, 6]$: good	$P \rightarrow$ good	0.87	$\mathcal{P}$
$[6, \mathcal{E}, 7]$: ,	$\mathcal{E} \rightarrow$,	-	-
$[8, P, 11]$: i still like it	$P \rightarrow$ i still like it	0.85	$\mathcal{P}$
$[4, P, 6]$: very good	$P \rightarrow$ very P	0.93	$\mathcal{P}$
$[3, N, 6]$: not very good	$N \rightarrow$ not P	0.63	$\mathcal{N}$
$[0, N, 6]$: the movie is not very good	$N \rightarrow PN$	0.60	$\mathcal{N}$
$[0, N, 7]$: the movie is not very good,	$N \rightarrow N\mathcal{E}$	0.60	$\mathcal{N}$
$[0, P, 11]$: the movie is not very good, but i still like it	$P \rightarrow N$ but P	0.76	$\mathcal{P}$
$[0, S, 11]$: the movie is not very good, but i still like it	$S \rightarrow P$	0.76	$\mathcal{P}$

The non-terminal set is denoted as $V_s = \{N, P, S, \mathcal{E}\}$, where S is the start symbol, the non-terminal N represents the negative polarity, and the non-terminal P represents the positive polarity. The rules in R_s are divided into the following six categories:

- *Dictionary rules*: $X \rightarrow w_0^k$, where $X \in \{N, P\}$, $w_0^k = w_0 \ldots w_{k-1}$, and $w_0^k \in \Sigma_s^+$. These rules can be regarded as the sentiment dictionary used in traditional approaches. They are basic sentiment units assigned with polarity probabilities. For instance, $P \rightarrow$ good is a dictionary rule.
- *Combination rules*: $X \rightarrow c$, where $c \in (V_s \cup \Sigma_s)^+$, and two successive non-terminals are not allowed. There is at least one terminal in c. These rules combine terminals and non-terminals, such as $N \rightarrow$ not P, and $P \rightarrow N$ but P. They are used to handle negation, intensification, and contrast in sentiment analysis. The number of non-terminals in a combination rule is restricted to one and two.
- *Glue rules*: $X \rightarrow X_1 X_2$, where $X, X_1, X_2 \in \{N, P\}$. These rules combine two text spans that are derived into X_1 and X_2, respectively.
- *OOV rules*: $\mathcal{E} \rightarrow w_0^k$, where $w_0^k \in \Sigma^+$. We use these rules to handle Out-Of-Vocabulary (OOV) text spans whose polarity probabilities are not learned from data.
- *Auxiliary rules*: $X \rightarrow \mathcal{E} X_1$, $X \rightarrow X_1 \mathcal{E}$, where $X, X_1 \in \{N, P\}$. These rules combine a text span with polarity and an OOV text span.
- *Start rules*: $S \rightarrow Y$, where $Y \in \{N, P, \mathcal{E}\}$. The derivations begin with S, and S can be derived to N, P, and $\mathcal{E}$.

Here, X represents the non-terminals N or P. The dictionary rules and combinations rules are automatically extracted from the data. We will describe the details in Section 4.2. By applying these rules, we can derive the polarity label of a sentence from the bottom–up. The glue rules are used to combine polarity information of two text spans together, and it treats the combined parts as independent. In order to tackle the OOV problem, we treat a text span that consists of OOV words as empty text span, and derive them to $\mathcal{E}$. The OOV text spans are combined with other text spans without considering their sentiment information. Finally, each sentence is derived to the symbol S using the start rules that are the beginnings of derivations. We can use the sentiment grammar to compactly describe the derivation process of a sentence.

3.2 Parsing Model

We present the formal description of the statistical sentiment parsing model following deductive proof systems (Shieber, Schabes, and Pereira 1995; Goodman 1999) as used in traditional syntactic parsing. For a concrete example,

$$\frac{(A \rightarrow BC) \quad [i, B, k] \quad [k, C, j]}{[i, A, j]} \tag{6}$$

which represents if we have the rule $A \rightarrow BC$ and $B \overset{*}{\Rightarrow} w_i^k$ and $C \overset{*}{\Rightarrow} w_k^j$ ($\overset{*}{\Rightarrow}$ is used to represent the reflexive and transitive closure of immediate derivation), then we can obtain $A \overset{*}{\Rightarrow} w_i^j$. By adding a unary rule

$$\frac{(A \rightarrow w_i^j)}{[i, A, j]} \tag{7}$$

with the binary rule in Equation (6), we can express the standard CYK algorithm for CFG in Chomsky Normal Form (CNF). And the goal is $[0, S, n]$, in which S is the start symbol and n is the length of the input sentence. In the given CYK example, the term in deductive rules can be one of the following two forms:

- $[i, X, j]$ is an **item** representing a subtree rooted in X spanning from i to j, or
- $(X \rightarrow \gamma)$ is a **rule** in the grammar.

Generally, we represent the form of an inference rule as:

$$\frac{(r) \quad H_1 \quad \dots \quad H_K}{[i, X, j]} \tag{8}$$

where, if all the terms r and H_k are true, then we can infer $[i, X, j]$ as true. Here, r denotes a sentiment rule, and H_k denotes an item. When we refer to both rules and items, we employ the word **terms**.

Theoretically, we can convert the sentiment rules to CNF versions, and then use the CYK algorithm to conduct parsing. Because the maximum number of non-terminal symbols in a rule is already restricted to two, we formulate the statistical sentiment parsing based on a customized CYK algorithm that is similar to the work of Chiang (2007). Let X, X_1, X_2 represent the non-terminals N or P; the inference rules for the statistical sentiment parsing are summarized in Figure 3.

3.3 Ranking Model

The parsing model generates many candidate parse trees $T(s)$ for a sentence s. The goal of the ranking model is to score and rank these parse trees. The sentiment tree with the highest score is treated as the best representation for sentence s. We extract a feature vector $\phi(s, t) \in \mathcal{R}^d$ for the specific sentence-tree pair (s, t), where $t \in T(s)$ is the parse tree. Let $\psi \in \mathcal{R}^d$ be the parameter vector for the features. We use the log-linear model to calculate a probability $p(t|s; T, \psi)$ for each parse tree $t \in T(s)$. The probabilities indicate how likely the trees are to produce correct predictions. Given the sentence s and parameters ψ, the log-linear model defines a conditional probability:

$$p(t|s; T, \psi) = \exp\left\{\phi(s, t)^{\mathsf{T}}\psi - \mathbf{A}(\psi; s, T)\right\} \tag{9}$$

$$\mathbf{A}(\psi; s, T) = \log \sum_{t \in T(s)} \exp\left\{\phi(s, t)^{\mathsf{T}}\psi\right\} \tag{10}$$

$$\frac{(X \to w_i^j)}{[i, X, j]}$$

$$\frac{(X \to w_i^{i_1} X_1 w_{j_1}^j) \quad [i_1, X_1, j_1]}{[i, X, j]}$$

$$\frac{(X \to w_i^{i_1} X_1 w_{j_1}^{i_2} X_2 w_{j_2}^j) \quad [i_1, X_1, j_1] \quad [i_2, X_2, j_2]}{[i, X, j]}$$

$$\frac{(X \to X_1 X_2) \quad [i, X_1, k] \quad [k, X_2, j]}{[i, X, j]}$$

$$\frac{(\mathcal{E} \to w_i^j)}{[i, \mathcal{E}, j]}$$

$$\frac{(X \to \mathcal{E} X_1) \quad [i, \mathcal{E}, k] \quad [k, X_1, j]}{[i, X, j]}$$

$$\frac{(X \to X_1 \mathcal{E}) \quad [i, X_1, k] \quad [k, \mathcal{E}, j]}{[i, X, j]}$$

where X, X_1, X_2 represent N or P.

Figure 3
Inference rules for the basic parsing model.

where $\mathbf{A}(\psi; s, T)$ is the log-partition function with respect to $T(s)$. The log-linear model is a discriminative model, and it is widely used in natural language processing. We can use $\phi(s, t)^{\mathsf{T}}\psi$ as the score of the parse tree without normalization in the decoding process, because $p(t|s; T, \psi) \propto \phi(s, t)^{\mathsf{T}}\psi$, and this will not change the ranking order.

3.4 Polarity Model

The goal of the polarity model is to model the calculation of sentiment strength and polarity of a text span from its subspans in the parsing process. It is specified in terms of the rules used in the parsing process. We expand the notations in the inference rule (8) to incorporate the polarity model. The new form of inference rule is:

$$\frac{(r) \quad H_1\Phi_1 \quad \dots \quad H_K\Phi_K}{[i, X, j]\Phi} \tag{11}$$

in which $r, H_1, \dots, H_K$ are the terms described in Section 3.2. Every item H_k is assigned polarity strength $\Phi_k : \begin{cases} P(\mathcal{N}|w_{i_k}^{j_k}) \\ P(\mathcal{P}|w_{i_k}^{j_k}) \end{cases}$ for text span $w_{i_k}^{j_k}$. For the item $[i, X, j]$, the polarity model $\Phi(r, \Phi_1, \dots, \Phi_K)$ is defined as a function that takes the rule r and polarity strength of subspans as input.

The polarity strength obtained by the polarity model should satisfy two constraints. First, the values calculated by the polarity model are non-negative, that is, $P(\mathcal{X}|w_i^j) \geq 0, P(\overline{\mathcal{X}}|w_i^j) \geq 0$. Second, the positive and negative polarity values are normalized to 1, namely, $P(\mathcal{X}|w_i^j) + P(\overline{\mathcal{X}}|w_i^j) = 1$. Notably, $\overline{\mathcal{X}} = \begin{cases} \mathcal{P}, & \mathcal{X} = \mathcal{N} \\ \mathcal{N}, & \mathcal{X} = \mathcal{P} \end{cases}$ is the opposite polarity of $\mathcal{X}$.

The inference rules with the polarity model are formally defined in Figure 4. In the following part, we define the polarity model for the different types of rules. If the rule is a dictionary rule $X \rightarrow w_i^j$, its sentiment strength is obtained as:

$$\Phi : \begin{cases} P(\mathcal{X}|w_i^j) = \tilde{P}(\mathcal{X}|w_i^j) \\ P(\overline{\mathcal{X}}|w_i^j) = \tilde{P}(\overline{\mathcal{X}}|w_i^j) \end{cases} \tag{12}$$

where $\mathcal{X} \in \{\mathcal{N}, \mathcal{P}\}$ denotes the sentiment polarity of the left hand side of the rule, $\overline{\mathcal{X}}$ is the opposite polarity of $\mathcal{X}$, and $\tilde{P}(\mathcal{X}|w_i^j), \tilde{P}(\overline{\mathcal{X}}|w_i^j)$ indicate the sentiment polarity values estimated from training data.

$$\frac{(X \rightarrow w_i^j)}{[i, X, j] P(\mathcal{X}|w_i^j) = \tilde{P}(\mathcal{X}|w_i^j)}$$

$$\frac{(X \rightarrow w_i^{i_1} X_1 w_{j_1}^j) \quad [i_1, X_1, j_1]\Phi_1}{[i, X, j] P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{X}_1|w_{i_1}^{j_1}))}$$

$$\frac{(X \rightarrow w_i^{i_1} X_1 w_{j_1}^{i_2} X_2 w_{j_2}^j) \quad [i_1, X_1, j_1]\Phi_1 \quad [i_2, X_2, j_2]\Phi_2}{[i, X, j] P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{X}_1|w_{i_1}^{j_1}) + \theta_2 P(\mathcal{X}_2|w_{i_2}^{j_2}))}$$

$$\frac{(X \rightarrow X_1 X_2) \quad [i, X_1, k]\Phi_1 \quad [k, X_2, j]\Phi_2}{[i, X, j] P(\mathcal{X}|w_i^j) = \frac{P(\mathcal{X}|w_i^k) P(\mathcal{X}|w_k^j)}{P(\mathcal{X}|w_i^k) P(\mathcal{X}|w_k^j) + P(\overline{\mathcal{X}}|w_i^k) P(\overline{\mathcal{X}}|w_k^j)}}$$

$$\frac{(\mathcal{E} \rightarrow w_i^j)}{[i, \mathcal{E}, j]\circ}$$

$$\frac{(X \rightarrow \mathcal{E} X_1) \quad [i, \mathcal{E}, k]\circ \quad [k, X_1, j]\Phi_1}{[i, X, j] P(\mathcal{X}|w_i^j) = P(\mathcal{X}|w_k^j)}$$

$$\frac{(X \rightarrow X_1 \mathcal{E}) \quad [i, X_1, k]\Phi_1 \quad [k, \mathcal{E}, j]\circ}{[i, X, j] P(\mathcal{X}|w_i^j) = P(\mathcal{X}|w_i^k)}$$

where $h(x) = \dfrac{1}{1 + \exp\{-x\}}$ is a logistic function, ○ represents the absence, and X, X_1, X_2 represent N or P. As specified in the polarity model, we have $P(\overline{\mathcal{X}}|w_i^j) = 1 - P(\mathcal{X}|w_i^j)$.

Figure 4
Inference rules with the polarity model.

The glue rules $X \to X_1 X_2$ combine two spans (w_i^k, w_k^j). The polarity value is calculated by their product, and normalized to 1.

$$\Phi : \begin{cases} P(\mathcal{X}|w_i^j) = \frac{P(\mathcal{X}|w_i^k)P(\mathcal{X}|w_k^j)}{P(\mathcal{X}|w_i^k)P(\mathcal{X}|w_k^j)+P(\overline{\mathcal{X}}|w_i^k)P(\overline{\mathcal{X}}|w_k^j)} \\ P(\overline{\mathcal{X}}|w_i^j) = 1 - P(\mathcal{X}|w_i^j) \end{cases} \tag{13}$$

For OOV text spans, the polarity model does not calculate the polarity values. When they are combined with in-vocabulary phrases by the auxiliary rules, the polarity values are determined by the text span with polarity and the OOV text span is ignored. More specifically,

$$\Phi : \begin{cases} P(\mathcal{X}|w_i^j) = P(\mathcal{X}|w_i^k) \\ P(\overline{\mathcal{X}}|w_i^j) = P(\overline{\mathcal{X}}|w_i^k) \end{cases} \tag{14}$$

The combination rules are more complicated than other types of rules. In this article, we model the polarity probability calculation as the logistic regression. The logistic regression can be regarded as putting linear combination of the subspans' polarity probabilities into a logistic function (or sigmoid function). We will show that the negation, intensification, and contrast can be well modeled by the regression-based method. It is formally shown as

$$\begin{aligned} P(\mathcal{X}|w_i^j) &= h\left(\theta_0 + \sum_{k=1}^{K} \theta_k P(\mathcal{X}_k|w_{i_k}^{j_k})\right) \\ &= \frac{1}{1 + \exp\left\{-\left(\theta_0 + \sum_{k=1}^{K} \theta_k P(\mathcal{X}_k|w_{i_k}^{j_k})\right)\right\}} \end{aligned} \tag{15}$$

where $h(x) = \frac{1}{1+\exp\{-x\}}$ is the logistic function, K is the number of non-terminals in a rule, and $\theta_0, \ldots, \theta_K$ are the parameters that are learned from data. As a concrete example, if the span w_i^j can match $N \to \text{not } P$ and $P \overset{*}{\Rightarrow} w_{i+1}^j$, the inference rule with the polarity model is defined as

$$\frac{N \to \text{not } P \quad [i+1, P, j]\Phi_1}{[i, N, j] \begin{cases} P(\mathcal{N}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{P}|w_{i+1}^j)) \\ P(\mathcal{P}|w_i^j) = 1 - P(\mathcal{N}|w_i^j) \end{cases}} \tag{16}$$

where polarity probability is calculated by $P(\mathcal{N}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{P}|w_{i+1}^j))$.

To tackle negation, **switch negation** (Choi and Cardie 2008; Saurí 2008) simply reverses the sentiment polarity and corresponding sentiment strength. However, consider *not great* and *not good*; flipping polarity directly makes *not good* more positive than *not great*, which is unreasonable. Another potential problem of switch negation is that negative polarity items interact with intensifiers in undesirable ways (Kennedy and Inkpen 2006). For example, *not very good* turns out to be even more negative than *not good*, given the fact that *very good* is more positive than *good*. Therefore, Taboada et al. (2011) argue that **shift negation** is a better way to handle polarity negation. Instead of reversing polarity strength, shift negation shifts it toward the opposite polarity by a fixed amount. This method can partially avoid the aforementioned two problems. However, they set the parameters manually, which might not be reliable and extensible enough to a new data set. Using the regression model, switch negation is captured by the negative scale item θ_k ($k > 0$), and shift negation is expressed by the shift item θ_0.

The intensifiers are adjectives or adverbs that strengthen (amplifier) or decrease (downtoner) the semantic intensity of its neighboring item (Quirk 1985). For example, *extremely good* should obtain higher strength of positive polarity than *good*, because it is modified by the amplifier (*extremely*). Polanyi and Zaenen (2006) and Kennedy and Inkpen (2006) handle intensifiers by polarity addition and subtraction. This method, termed **fixed intensification**, increases a fixed amount of polarity for amplifiers and decreases for downtoners. Taboada et al. (2011) propose a method, called **percentage intensification**, to associate each intensification word with a percentage scale, which is larger than one for amplifiers, and less than one for downtoners. The regression model can capture these two methods to handle the intensification. The shift item θ_0 represents the polarity addition and subtraction directly, and the scale item θ_k ($k > 0$) can scale the polarity by a percentage.

Table 2 illustrates how the regression based polarity model represents different negation and intensification methods. For a specific rule, the parameters and the compositional method are automatically learned from data (Section 4.2.3) instead of setting them manually as in previous work (Taboada et al. 2011). In a similar way, this method can handle the contrast. For example, the inference rule for *N* → *P but N* is:

$$\frac{(N \rightarrow P \text{ but } N) \quad [i_1, P, j_1]\Phi_1 \quad [i_2, N, j_2]\Phi_2}{[i, N, j] \begin{cases} P(\mathcal{N}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{P}|w_{i_1}^{j_1}) + \theta_2 P(\mathcal{N}|w_{i_2}^{j_2})) \\ P(\mathcal{P}|w_i^j) = 1 - P(\mathcal{N}|w_i^j) \end{cases}} \tag{17}$$

where the polarity probability of the rule *N* → *P but N* is computed by $P(\mathcal{N}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{P}|w_{i_1}^{j_1}) + \theta_2 P(\mathcal{N}|w_{i_2}^{j_2}))$. It can express the contrast relation by specific parameters θ_0, θ_1, and θ_2.

It should be noted that a linear regression model could turn out to be problematic, as it may produce unreasonable results. For example, if we do not add any constraint, we may get $P(\mathcal{N}|w_i^j) = -0.6 + P(\mathcal{P}|w_{i+1}^j)$. When $P(\mathcal{P}|w_{i+1}^j) = 0.55$, we will get $P(\mathcal{N}|w_i^j) = -0.6 + 0.55 = -0.05$. This conflicts with the definition that the polarity probability ranges from zero to one. Figure 5 intuitively shows that the logistic function truncates polarity values to $(0, 1)$ smoothly.

Table 2
The check mark means the parameter of the polarity model can capture the corresponding intensification type and negation type. Shift item θ_0 can handle shift negation and fixed intensification, and scale item θ_1 can model switch negation and percentage intensification.

Parameter	Negation Type $P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\overline{\mathcal{X}}|w_{i_1}^{j_1}))$		Intensification Type $P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{X}|w_{i_1}^{j_1}))$	
	Shift	Switch	Percentage	Fixed
θ_0 (Shift item)	✓			✓
θ_1 (Scale item)		✓	✓	

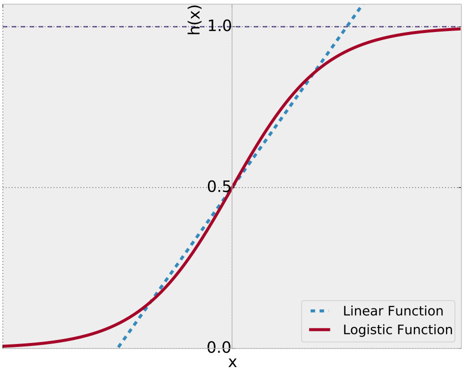


Figure 5
Logistic function $h(x) = \frac{1}{1+\exp\{-x\}}$ truncates polarity values to $(0, 1)$ smoothly. The computed values are used as polarity probabilities.

3.5 Constraints

We incorporate additional constraints into the parsing model. Those are used as pruning conditions in the derivation process not only to improve efficiency but also to force the derivation towards the correct direction. We expand the inference rules in Section 3.4 as,

$$\frac{(r) \quad H_1\Phi_1 \quad \ldots \quad H_K\Phi_K}{[i, X, j]\Phi} C \tag{18}$$

where C is a **side condition**. The constraints are interpreted in a Boolean manner. If the constraint C is satisfied, the rule can be used, otherwise, it cannot. We define two constraints in the parsing model.

First, in the parsing process, the polarity label of text span w_i^j obtained by the polarity model (Section 3.4) should be consistent with the non-terminal X (N or P) on the left hand side of the rule. To distinguish between the polarity labels and the non-terminals, we denote the corresponding polarity label of non-terminal X as $\mathcal{X}$. Following this notation, we describe the first constraint as

$$C_1 : P(\mathcal{X}|w_i^j) > P(\overline{\mathcal{X}}|w_i^j) \tag{19}$$

where $\overline{\mathcal{X}}$ is the opposite polarity of $\mathcal{X}$. For instance, if rule $P \rightarrow$ *not* N matches the text span w_i^j, the polarity calculated by the polarity model should be consistent with P, i.e., the polarity obtained by the polarity model should be positive ($\mathcal{P}$).

Second, when we apply the combination rules, the polarity strength of subspans needs to exceed a predefined threshold τ (≥ 0.5). Specifically, for combination rules $X \rightarrow w_i^{i_1} X_1 w_{j_1}^{i_2} X_2 w_{j_2}^{j}$ and $X \rightarrow w_i^{i_1} X_1 w_{j_1}^{j}$, we define the second constraint as

$$C_2 : P(\mathcal{X}_k|w_{i_k}^{j_k}) > \tau, k = 1, \ldots, K \tag{20}$$

where K is the number of subspans in the rule, and $\mathcal{X}_k$ is the corresponding polarity label of non-terminal X_k in the right hand side. If $P(\mathcal{X}_k|w_{i_k}^{j_k})$ is not larger than threshold

τ, we regard the polarity of phrase $w_{i_k}^{j_k}$ as neutral. For instance, we do not want to use the combination rule $P \rightarrow$ a lot of P or $N \rightarrow$ a lot of N for the phrase *a lot of people*. This constraint avoids improperly using the combination rules for neutral phrases. Notably, when τ is set as 0.5, this constraint is the same as the first one in Equation (19).

As shown in Figure 6, we add these two constraints to the inference rules. The OOV rules do not have any constraints, and the constraint C_1 is applied for all the other rules. The constraint C_2 is only applied for the combination rules.

3.6 Decoding Algorithm

In this section, we summarize the decoding algorithm in Algorithm 1. For a sentence s, the CYK algorithm and dynamic programming are used to obtain the sentiment tree with the highest score. To be specific, the modified CYK parsing model parses the input sentence to sentiment trees in a bottom–up manner—that is, from short to long text spans. For every text span w_i^j, we match the rules in the sentiment grammar (Section 3.1) to generate the candidate set. Their polarity values are calculated using the polarity model described in Section 3.4. We also use the constraints described in Section 3.5 to prune search paths. The constraints improve the efficiency of the parsing algorithm and make derivations that meet our intuition.

$$\frac{(X \rightarrow w_i^j)}{[i,X,j]P(\mathcal{X}|w_i^j) = \tilde{P}(\mathcal{X}|w_i^j)} C_1$$

$$\frac{(X \rightarrow w_i^{i_1} X_1 w_{j_1}^j) \quad [i_1, X_1, j_1]\Phi_1}{[i,X,j]P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{X}_1|w_{i_1}^{j_1}))} C_1 \wedge C_2$$

$$\frac{(X \rightarrow w_i^{i_1} X_1 w_{j_1}^{i_2} X_2 w_{j_2}^j) \quad [i_1, X_1, j_1]\Phi_1 \quad [i_2, X_2, j_2]\Phi_2}{[i,X,j]P(\mathcal{X}|w_i^j) = h(\theta_0 + \theta_1 P(\mathcal{X}_1|w_{i_1}^{j_1}) + \theta_2 P(\mathcal{X}_2|w_{i_2}^{j_2}))} C_1 \wedge C_2$$

$$\frac{(X \rightarrow X_1 X_2) \quad [i, X_1, k]\Phi_1 \quad [k, X_2, j]\Phi_2}{[i,X,j]P(\mathcal{X}|w_i^j) = \frac{P(\mathcal{X}|w_i^k)P(\mathcal{X}|w_k^j)}{P(\mathcal{X}|w_i^k)P(\mathcal{X}|w_k^j) + P(\overline{\mathcal{X}}|w_i^k)P(\overline{\mathcal{X}}|w_k^j)}} C_1$$

$$\frac{(\mathcal{E} \rightarrow w_i^j)}{[i,\mathcal{E},j]\circ} \circ$$

$$\frac{(X \rightarrow \mathcal{E} X_1) \quad [i,\mathcal{E},k] \circ \quad [k, X_1, j]\Phi_1}{[i,X,j]P(\mathcal{X}|w_i^j) = P(\mathcal{X}|w_k^j)} C_1$$

$$\frac{(X \rightarrow X_1 \mathcal{E}) \quad [i, X_1, k]\Phi_1 \quad [k,\mathcal{E},j]\circ}{[i,X,j]P(\mathcal{X}|w_i^j) = P(\mathcal{X}|w_i^k)} C_1$$

where $h(x) = \frac{1}{1+\exp\{-x\}}$ is a logistic function, $\circ$ represents the absence, and X, X_1, X_2 represent N or P. As specified in the polarity model, we have $P(\overline{\mathcal{X}}|w_i^j) = 1 - P(\mathcal{X}|w_i^j)$.

Figure 6
Inference rules with the polarity model and constraints.

Algorithm 1 Decoding Algorithm

```
Input: $w_0^n$: Sentence
Output: Polarity of the input sentence
$score[,,] \leftarrow \{\}$
for $l \leftarrow 1 \dots n$ do                                    ▷ Modified CYK algorithm
    for all $i, j$ $s.t.$ $j - i = l$ do
        for all inferable rule $\frac{(r)\ H_1 \dots H_K}{[i,X,j]}$ for $w_i^j$ do
            $\Phi \leftarrow$ calculate polarity value for $r$                ▷ Polarity model
            if constraints are satisfied then                          ▷ Constraint
                $sc \leftarrow$ compute score for this derivation by ranking model    ▷ Ranking model
                if $sc > score[i, j, X]$ then
                    $score[i, j, X] \leftarrow sc$
return $\arg\max_{\mathcal{X} \in \{\mathcal{N},\mathcal{P}\}} score[0, X, n]$
```

The features in the ranking model (Section 4.1.1) decompose along the structure of the sentiment tree. So the dynamic programming technique can be used to compute the derivation tree with the highest ranking score. For a span, the scores of its subspans are used to calculate the local scores of its derivations. For example, the score of the derivation $\frac{(r)\ [i_1,P,j_1]\ [i_2,N,j_2]}{[i,X,j]}$ is $score[i_1, j_1, P] + score[i_2, j_2, N] + score^r$, where $score[i, j, X]$ is the highest score of text span w_i^j that is derived to the non-terminal X, and $score^r$ is the score of applying the rule r. As described in Section 3.3, the score of using rule r is $score^r = \phi(w_i^j, r)^\intercal \psi$, where $\phi(w_i^j, r)$ is the feature vector of using the rule r for the span w_i^j, and ψ is the weight vector of the ranking model. The k highest score trees satisfying the constraints are stored in $score[,,]$ for decoding the longer text spans. After finishing the CYK parsing, $\arg\max_{\mathcal{X} \in \{\mathcal{N},\mathcal{P}\}} score[0, n, X]$ is regarded as the polarity label of input sentence. The time complexity is the same as the standard CYK's.

4. Model Learning

We described the statistical sentiment parsing framework in Section 3. We present the model learning process in this section. The learning process consists of two steps. First, the sentiment grammar and the polarity model are learned from data. In other words, the rules and the parameters used to compute polarity values are learned. These basic sentiment building blocks are then used to build the parse trees. Second, we estimate the parameters of the ranking model using the sentence and polarity label pairs. At this stage, we concentrate on learning how to score the parse trees based on the learned sentiment grammar and polarity model.

Section 4.1 shows the features and the parameter estimation algorithm used in the ranking model. Section 4.2 illustrates how to learn the sentiment grammar and the polarity model.

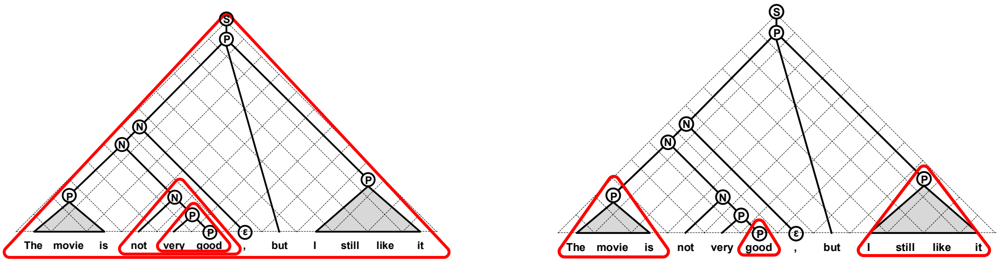
4.1 Ranking Model Training

As shown in Section 3.3, we develop the ranking model on the log-linear model. In the following subsections, we first present the features used to rank sentiment tree candidates. Then, we describe the objective function used in the optimization algorithm. Finally, we introduce the algorithm for parameter estimation using the gradient-based method.

4.1.1 Features. We extract a feature vector $\phi(s,t) \in \mathcal{R}^d$ for each parse tree t of sentence s. The feature vector is used in the log-linear model. In Figure 7, we present the features extracted for the sentence *The movie is not very good, but i still like it*. The features are organized into feature templates. Each of them contains a set of features. These feature templates are shown as follows:

- CombHit: This feature is the total number of combination rules used in t.
- CombRule: It contains features $\{$CombRule$[r] : r$ is a combination rule$\}$, each of which fires on the combination rule r appearing in t.
- DictHit: This feature is the total number of dictionary rules used in t.
- DictRule: It contains features $\{$DictRule$[r] : r$ is a dictionary rule$\}$, each of which fires on the dictionary rule r appearing in t.

These features are generic local patterns that capture the properties of the sentiment tree. Another intuitive lexical feature template is [combination rule + word]. For instance, $P \rightarrow$ very P(good) is a feature that lexicalizes the non-terminal P to good. However, if this feature is fired frequently, the phrase *very good* would be learned as a dictionary rule and can be used in the decoding process. So we do not use this feature template in order to reduce the feature size. It should be noted that these features



(a) CombHit and CombRule

(b) DictHit and DictRule

Feature Template	Feature	Feature Value
Number of combination rules	CombHit	3
Combination rule	CombRule[$P \rightarrow$ very P]	1
	CombRule[$N \rightarrow$ not P]	1
	CombRule[$P \rightarrow N$ but P]	1
Number of dictionary rules	DictHit	3
Dictionary rule	DictRule[$P \rightarrow$ the movie is]	1
	DictRule[$P \rightarrow$ good]	1
	DictRule[$P \rightarrow$ i still like it]	1

Figure 7
Feature templates used in the ranking model. The red triangles denote the features for the example.

decompose along structures of sentiment trees, enabling us to use dynamic programming in the CYK algorithm.

4.1.2 Objective Function. We design the ranking model upon the log-linear model to score candidate sentiment trees. In the training data $\mathcal{D}$, we only have the input sentence s and its polarity label $\mathcal{L}_s$. The forms of sentiment parse trees, which can obtain the correct sentiment polarity, are unobserved. So we work with the marginal log-likelihood of obtaining the correct polarity label $\mathcal{L}_s$,

$$\begin{aligned}\log p(\mathcal{L}_s|s;T,\psi) &= \log p(t \in T^{\mathcal{L}_s}(s)|s;T,\psi) \\ &= \mathbf{A}(\psi;s,T^{\mathcal{L}_s}) - \mathbf{A}(\psi;s,T)\end{aligned} \tag{21}$$

where $T^{\mathcal{L}_s}$ is the set of candidate trees whose prediction labels are $\mathcal{L}_s$, and $\mathbf{A}(\psi;s,T)$ (Equation (10)) is the log-partition function with respect to $T(s)$.

Based on the marginal log-likelihood function, the objective function $\mathcal{O}(\psi,T)$ consists of two terms. The first term is the sum of marginal log-likelihood over training instances that can obtain the correct polarity labels. The second term is a L^2-norm regularization term on the parameters ψ. Formally,

$$\mathcal{O}(\psi,T) = \sum_{\substack{(s,\mathcal{L}_s)\in\mathcal{D} \\ T^{\mathcal{L}_s}(s)\neq\emptyset}} \log p(\mathcal{L}_s|s;T,\psi) - \frac{\lambda}{2}||\psi||_2^2 \tag{22}$$

To learn the parameters ψ, we use a gradient-based optimization method to maximize the objective function $\mathcal{O}(\psi,T)$. According to Wainwright and Jordan (2008), the derivative of the log-partition function is the expected feature vector

$$\frac{\partial\mathcal{O}(\psi,T)}{\partial\psi} = \sum_{\substack{(s,\mathcal{L}_s)\in\mathcal{D} \\ T^{\mathcal{L}_s}(s)\neq\emptyset}} (E_{p(t|s;T^{\mathcal{L}_s},\psi)}[\phi(s,t)] - E_{p(t|s;T,\psi)}[\phi(s,t)]) - \lambda\psi \tag{23}$$

where $E_{p(x)}[f(x)] = \sum_x p(x)f(x)$ for discrete x.

4.1.3 Parameter Estimation. The objective function $\mathcal{O}(\psi,T)$ is not concave (nor convex), hence the optimization potentially results in a local optimum. Stochastic Gradient Descent (SGD; Robbins and Monro 1951) is a widely used optimization method. The SGD algorithm picks up a training instance randomly, and updates the parameter vector ψ according to

$${\psi_j}^{(t+1)} = {\psi_j}^{(t)} + \alpha\left(\frac{\partial\mathcal{O}(\psi)}{\partial\psi_j}|_{\psi=\psi^{(t)}}\right) \tag{24}$$

where α is the learning rate, and $\frac{\partial\mathcal{O}(\psi)}{\partial\psi_j}$ is the gradient of the objective function with respect to parameter ψ_j. The SGD is sensitive to α, and the learning rate is the same for all dimensions. As described in Section 4.1.1, we mix sparse features together with dense features. We want the learning rate to be different for each dimension. We use AdaGrad (Duchi, Hazan, and Singer 2011) to update the parameters, which sets an adaptive per-feature learning rate. The AdaGrad algorithm tends to use smaller update steps when we meet a feature many times. In order to compute efficiently, a diagonal approximation version of AdaGrad is used. The update rule is

$$\begin{aligned}\psi_j^{(t+1)} &= \psi_j^{(t)} + \alpha \frac{1}{\sqrt{G_j^{(t+1)}}} \left(\frac{\partial \mathcal{O}(\psi)}{\partial \psi_j} |_{\psi=\psi^{(t)}} \right) \\ G_j^{(t+1)} &= G_j^{(t)} + \left(\frac{\partial \mathcal{O}(\psi)}{\partial \psi_j} |_{\psi=\psi^{(t)}} \right)^2\end{aligned} \tag{25}$$

where we introduce an adaptive term $G_j^{(t)}$. $G_j^{(t)}$ becomes larger along with updating, and decreases the update step for dimension j. Compared with SGD, the only cost is to store and update $G_j^{(t)}$ for each parameter.

To train the model, we use the method proposed by Liang, Jordan, and Klein (2013). With the candidate parse trees and objective function, the parameters ψ are updated to make the parsing model favor correct trees and give them a higher score. Because there are many parse trees for a sentence, we need to calculate Equation (23) efficiently. As indicated in Section 4.1.1, the features decompose along the structure of sentiment trees. So dynamic programming can be utilized to compute $E_{p(t|s;T,\psi)}[\phi(s,t)]$ of Equation (23). However, the first expectation term $E_{p(t|s;T^{\mathcal{L}_s},\psi)}[\phi(s,t)]$ sums over the candidates that obtain the correct polarity labels. As this constraint does not decompose along the tree structure, there is no efficient dynamic program for this. Instead of searching all the parse trees spanning s, we use beam search to approximate this expectation. Beam search is a best-first search algorithm that explores at most K paths (K is the beam size). It keeps the local optimums to reduce the huge search space. Specifically, the beam search algorithm generates the K-best trees with the highest score $\phi(s,t)^{\mathsf{T}}\psi$ for each span. These local optimums are used recursively in the CYK process. The K-best trees for the whole span are regarded as the candidate set $\tilde{T}$. Then $\tilde{T}$ and $\tilde{T}^{\mathcal{L}_s}$ are used to approximate Equation (23) as in Liang, Jordan, and Klein (2013).

The intuition behind this parameter estimation algorithm lies in: (1) if we have better parameters, we can obtain better candidate trees; (2) with better candidate trees, we can learn better parameters. Thus the optimization problem is solved in an iterative manner. We initialize the parameters as zeros. This leads to a random search and generates random candidate trees. With the initial candidates, the two steps in Algorithm 2 lead the parameters ψ towards the direction achieving better performance.

4.2 Sentiment Grammar Learning

In this section, we present the automatic learning of the sentiment grammar as defined in Section 3.1. We need to extract the dictionary rules and the combination rules from data. In traditional statistical parsing, grammar rules are induced from annotated parse trees (such as the Penn TreeBank), so ideally we need examples of sentiment structure trees, or sentences annotated with sentiment polarity for the whole sentence as well as those for constituents within sentences. However, this is not practical, if not unfeasible, as the annotations will be inevitably time consuming and require laborious human effort. We show that it is possible to induce the sentiment grammar directly from examples of sentences annotated with sentiment polarity labels without using any syntactic annotations or polarity annotations of constituents within sentences. The

Algorithm 2 Ranking Model Learning Algorithm

```
Input: $\mathcal{D}$: Training data $\{(s, \mathcal{L}_s)\}$, $S$: Maximum number of iteration
Output: $\psi$: Parameters of the ranking model
$\psi^{(0)} \leftarrow (0, 0, \dots, 0)^{\mathsf{T}}$
repeat
    $(s, \mathcal{L}_s) \leftarrow$ randomly select a training instance in $\mathcal{D}$
    $\tilde{T}^{(t)} \leftarrow$ BEAMSEACH$(s, \psi^{(t)})$    ▷ Beam search to generate K-best candidates
    $G_j^{(t+1)} \leftarrow G_j^{(t)} + \left(\frac{\partial \mathcal{O}(\psi, \tilde{T}^{(t)})}{\partial \psi_j}|_{\psi=\psi^{(t)}}\right)^2$
    $\psi_j^{(t+1)} \leftarrow \psi_j^{(t)} + \alpha \frac{1}{\sqrt{G_j^{(t+1)}}} \left(\frac{\partial \mathcal{O}(\psi, \tilde{T}^{(t)})}{\partial \psi_j}|_{\psi=\psi^{(t)}}\right)$    ▷ Update parameters using AdaGrad
    $t \leftarrow t + 1$
until $t > S$
return $\psi^{(T)}$
```

sentences annotated with sentiment polarity labels are relatively easy to obtain, and we use them as our input to learn dictionary rules and combination rules.

We first present the basic idea behind the algorithm we propose. People are likely to express positive or negative opinions using very simple and straightforward sentiment expressions again and again in their reviews. Intuitively, we can mine dictionary rules from these massive review sentences by leveraging the redundancy characteristics. Furthermore, there are many complicated reviews that contain complex sentiment structures (e.g., negation, intensification, and contrast). If we already have dictionary rules on hand, we can use them to obtain basic sentiment information for the fragments within complicated reviews. We can then extract combination rules with the help of the dictionary rules and the sentiment polarity labels of complicated reviews. Because the simple and straightforward sentiment expressions are often coupled with complicated expressions, we need to conduct dictionary rule mining and the combination rule mining in an iterative way.

4.2.1 Dictionary Rule Learning. The dictionary rules $\mathcal{G}_{\mathcal{D}}$ are basic sentiment building blocks used in the parsing process. Each dictionary rule in $\mathcal{G}_{\mathcal{D}}$ is in the form $X \rightarrow f$, where f is a sentiment fragment. We use the polarity probabilities $P(\mathcal{N}|f)$ and $P(\mathcal{P}|f)$ in the polarity model. To build $\mathcal{G}_{\mathcal{D}}$, we regard all the frequent fragments whose occurrence frequencies are larger than τ_f and lengths range from 1 to 7 as the sentiment fragments. We further filter the phrases formed by stop words and punctuations, which are not used to express sentiment.

For a balanced data set, the sentiment distribution of a candidate sentiment fragment f is calculated by

$$P(\mathcal{X}|f) = \frac{\#(f, \mathcal{X}) + 1}{\#(f, \mathcal{N}) + \#(f, \mathcal{P}) + 2} \tag{26}$$

where $\mathcal{X} \in \{\mathcal{N}, \mathcal{P}\}$, and $\#(f, \mathcal{X})$ denotes the number of reviews containing f with $\mathcal{X}$ being the polarity. It should be noted that Laplace smoothing is used in Equation (26) to deal with the zero frequency problem.

We do not learn the polarity probabilities $P(\mathcal{N}|f)$ and $P(\mathcal{P}|f)$ by directly counting occurrence frequency. For example, in the review sentence *this movie is not good* (negative), the naive counting method increases the count #(*good*, $\mathcal{N}$) in terms of the polarity of the whole sentence. Moreover, because of the common collocation *not as good as* (negative) in movie reviews, *as good as* is also regarded as negative if we count the frequency directly. The examples indicate why some polarity probabilities of phrases counting from data are different from our intuitions. These unreasonable polarity probabilities also make trouble for learning the polarity model. Consequently, in order to estimate more reasonable probabilities, we need to take the compositionality into consideration when learning sentiment fragments.

Following this motivation, we ignore the count $\#(f, \mathcal{X})$, if the sentiment fragment f is **covered** by a negation rule r that negates the polarity of f. The word **cover** here means that f is derived within a non-terminal of the negation rule r. For instance, the negation rule $N \rightarrow$ not P covers the sentiment fragment *good* in the sentence *this is not a good movie* (negative), that is, the *good* is derived from P of this negation rule. So we ignore the occurrence for #(good, $\mathcal{N}$) in this sentence. It should be noted that we still increase the count for #(not good, $\mathcal{N}$), because there is no negation rule covering the fragment *not good*.

As shown in Algorithm 3, we learn the dictionary rules and their polarity probabilities by counting the frequencies in negative and positive classes. Only the fragments whose occurrence numbers are larger than threshold τ_f are kept. Moreover, we take the combination rules into consideration to acquire more reasonable $\mathcal{G}_{\mathcal{D}}$. Notably, a subsequence of a frequent fragment must also be frequent. This is similar to the key insight in the Apriori algorithm (Agrawal and Srikant 1994). When we learn the dictionary rules, we can count the sentiment fragments from short to long, and prune the infrequent fragments in the early stages if any subsequence is not frequent. This pruning method accelerates the dictionary rule learning process and makes the procedure fit in memory.

Algorithm 3 Dictionary Rule Learning

```
Input: $\mathcal{D}$: Data set, $\mathcal{G}_{\mathcal{C}}$: Combination rules, $\tau_f$: Frequency threshold
Output: $\mathcal{G}_{\mathcal{D}}$: Dictionary rules
function MINEDICTIONARYRULES($\mathcal{D}, \mathcal{G}_{\mathcal{C}}$)
    ${\mathcal{G}_{\mathcal{D}}}' \leftarrow \{\}$
    for $(s, \mathcal{L}_s)$ in $\mathcal{D}$ do                    ▷ $s : w_0 w_1 \cdots w_{|s|-1}$, $\mathcal{L}_s$: Polarity label of $s$
        for all $i, j$ $s.t.$ $0 \leq i < j \leq |s|$ do               ▷ $w_i^j : w_i w_{i+1} \cdots w_{j-1}$
            if no negation rule in $\mathcal{G}_{\mathcal{C}}$ covers $w_i^j$ then
                $\#(w_i^j, \mathcal{L}_s) ++$
            add $w_i^j$ to ${\mathcal{G}_{\mathcal{D}}}'$
    $\mathcal{G}_{\mathcal{D}} \leftarrow \{\}$
    for $f$ in ${\mathcal{G}_{\mathcal{D}}}'$ do
        if $\#(f, \cdot) \geq \tau_f$ then
            compute $P(\mathcal{N}|f)$ and $P(\mathcal{P}|f)$ using Equation (26)
            add dictionary rule $(L_f \rightarrow f)$ to $\mathcal{G}_{\mathcal{D}}$        ▷ $L_f = \arg\max_{X \in \{N,P\}} P(\mathcal{X}|f)$
    return $\mathcal{G}_{\mathcal{D}}$
```

4.2.2 Combination Rule Learning. The combination rules $\mathcal{G}_C$ are generalizations for the dictionary rules. They are used to handle the compositionality and process unseen phrases. The learning of combination rules is based on the learned dictionary rules and their polarity values. The sentiment fragments are generalized to combination rules by replacing the subsequences of dictionary rules with their polarity labels. For instance, as shown in Figure 8, the fragments *is not (good/as expected/funny/well done)* are all negative. After replacing the subspans *good*, *as expected*, *funny*, and *well done* with their polarity label P, we can learn the negation rule $N \rightarrow$ is not P.

We present the combination rule learning approach in Algorithm 4. Specifically, the first step is to generate combination rule candidates. For every subsequence w_i^j of sentiment fragment f, we replace it with the corresponding non-terminal $L_{w_i^j}$ if $P(\mathcal{L}_{w_i^j}|w_i^j)$ is larger than the threshold τ_p, and we can get $w_0^i L_{w_i^j} w_j^{|f|}$. Next, we compare the polarity $\mathcal{L}_{w_i^j}$ with $\mathcal{L}_f$. If $\mathcal{L}_f \neq \mathcal{L}_{w_i^j}$, we regard the rule $L_f \rightarrow w_0^i L_{w_i^j} w_j^{|f|}$ as a negation rule. Otherwise, we further compare their polarity values. If this rule makes the polarity value become larger (or smaller), it will be treated as a strengthen (or weaken) rule. To obtain the contrast rules, we replace two subsequences with their polarity labels in a similar way. If the polarities of these two subsequences are different, we categorize this rule to the contrast type. Notably, these two non-terminals cannot be next to each other. After these steps, we get the rule candidate set $\mathcal{G}_C{}'$ and the occurrence number of each rule. We then filter the rule candidates whose occurrence frequencies are too small, and assign the rule types (negation, strengthen, weaken, and contrast) according to their occurrence numbers.

4.2.3 Polarity Model Learning. As shown in Section 3.4, we define the polarity model to calculate the polarity probabilities using the sentiment grammar. In this section, we present how to learn the parameters of the polarity model for the combination rules.

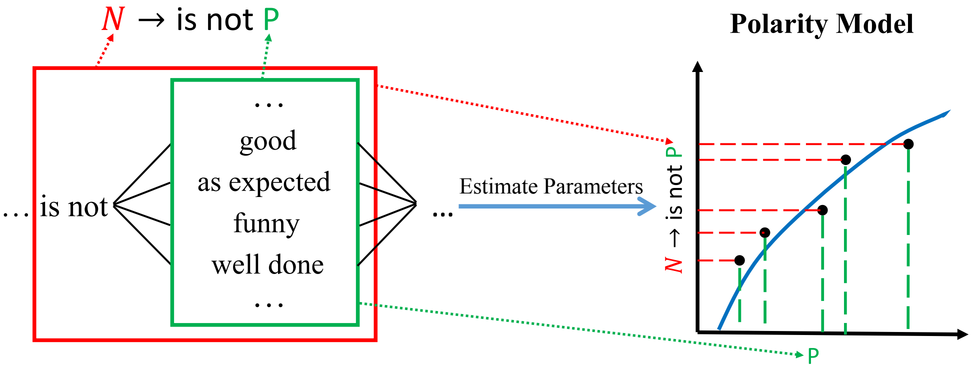


Figure 8
We replace the subsequences with their polarity labels for frequent sentiment fragments. As shown here, we replace *good*, *as expected*, *funny*, and *well done* with their polarity label P. Then we compare the polarity probabilities of subfragments with the whole fragments, such as *good* and *is not good*, to determine whether it is a negation rule, strengthen rule, or weaken rule. After obtaining the rule, we use polarity probabilities of these compositional examples as training data to estimate parameters of the polarity model. In this, $\big(P(\mathcal{P}|\text{good}), P(\mathcal{N}|\text{is not good})\big)$, $\big(P(\mathcal{P}|\text{as expected}), P(\mathcal{N}|\text{is not as expected})\big)$, $\big(P(\mathcal{P}|\text{funny}), P(\mathcal{N}|\text{is not funny})\big)$, $(P(\mathcal{P}|\text{well done}), \text{and } P(\mathcal{N}|\text{is not well done}))$ are used to learn the polarity model for $N \rightarrow$ is not P.

Algorithm 4 Combination Rule Learning

```
Input: $\mathcal{D}$: Data set, $\mathcal{G}_\mathcal{D}$: Dictionary rules, $\tau_p, \tau_\Delta, \tau_r, \tau_c$: Thresholds
Output: $\mathcal{G}_C$: Combination rules
function MINECOMBINATIONRULES($\mathcal{D}, \mathcal{G}_\mathcal{D}$)
    $\mathcal{G}_C{}' \leftarrow \{\}$
    for $(X \rightarrow f)$ in $\mathcal{G}_\mathcal{D}$ do                              ▷ $f : w_0 w_1 \cdots w_{|f|-1}$
        for all $i, j$ $s.t.$ $0 \leq i < j \leq |f|$ do
            if $P(\mathcal{L}_{w_i^j}|w_i^j) > \tau_p$ then        ▷ Polarity label $\mathcal{L}_{w_i^j} = \arg\max_{\mathcal{X}\in\{\mathcal{N},\mathcal{P}\}} P(\mathcal{X}|w_i^j)$
                $r$: $X \rightarrow w_0^i L_{w_i^j} w_j^{|f|}$            ▷ Non-terminal $L_{w_i^j} = \arg\max_{X\in\{N,P\}} P(\mathcal{X}|w_i^j)$
                if $\mathcal{X} \neq \mathcal{L}_{w_i^j}$ then
                    $\#(r, negation)$ ++
                else if $P(\mathcal{X}|f) > P(\mathcal{L}_{w_i^j}|w_i^j) + \tau_\Delta$ then
                    $\#(r, strengthen)$ ++
                else if $P(\mathcal{X}|f) < P(\mathcal{L}_{w_i^j}|w_i^j) - \tau_\Delta$ then
                    $\#(r, weaken)$ ++
                add $r$ to $\mathcal{G}_C{}'$
        for all $i_0, j_0, i_1, j_1$ $s.t.$ $0 \leq i_0 < j_0 < i_1 < j_1 \leq |f|$ do
            if $P(\mathcal{L}_{w_{i_0}^{j_0}}|w_{i_0}^{j_0}) > \tau_p$ and $P(\mathcal{L}_{w_{i_1}^{j_1}}|w_{i_1}^{j_1}) > \tau_p$ then
                $r$: $X \rightarrow w_0^{i_0} L_{w_{i_0}^{j_0}} w_{j_0}^{i_1} L_{w_{i_1}^{j_1}} w_{j_1}^{|f|}$      ▷ Replace $w_{i_0}^{j_0}$, $w_{i_1}^{j_1}$ with the non-terminals
                if $\mathcal{L}_{w_{i_0}^{j_0}} \neq \mathcal{L}_{w_{i_1}^{j_1}}$ then
                    $\#(r, contrast)$ ++
                add $r$ to $\mathcal{G}_C{}'$
    $\mathcal{G}_C \leftarrow \{\}$
    for $r$ in $\mathcal{G}_C{}'$ do
        if $\#(r, \cdot) > \tau_r$ and $\max_T \frac{\#(r,T)}{\#(r)} > \tau_c$ then
            add $r$ to $\mathcal{G}_C$
    return $\mathcal{G}_C$
```

As shown in Figure 8, we learn combination rules by replacing the subsequences of frequent sentiment fragments with their polarity labels. Both the replaced fragment and the whole fragment can be found in the dictionary rules, so their polarity probabilities have been estimated from data. We can use them as our training examples to figure out how context changes the polarity of replaced fragment, and learn parameters of the polarity model.

We describe the polarity model in Section 3.4. To further simplify the notation, we denote the input vector $\mathbf{x} = (1, P(\mathcal{X}_1|w_{i_1}^{j_1}), \ldots, P(\mathcal{X}_K|w_{i_K}^{j_K}))^\intercal$, and the response value as y. Then we can rewrite Equation (15) as

$$h_\theta(\mathbf{x}) = \frac{1}{1 + \exp\{-\theta^\intercal \mathbf{x}\}} \tag{27}$$

where $h_\theta(\mathbf{x})$ is the polarity probability calculated by the polarity model, and $\theta = (\theta_0, \theta_1, \ldots, \theta_K)^\mathsf{T}$ is the parameter vector. Our goal is to estimate the parameter vector θ of the polarity model.

We fit the model to minimize the sum of squared residuals between the predicted polarity probabilities and the values computed from data. We define the cost function as

$$\mathcal{J}(\theta) = \frac{1}{2} \sum_m \left(h_\theta(\mathbf{x}^m) - y^m\right)^2 \tag{28}$$

where $\left(\mathbf{x}^m, y^m\right)$ is the m-th training instance.

The gradient descent algorithm is used to minimize the cost function $\mathcal{J}(\theta)$. The partial derivative of $\mathcal{J}(\theta)$ with respect to θ_j is

$$\begin{aligned}\frac{\partial \mathcal{J}(\theta)}{\partial \theta_j} &= \sum_m \left(h_\theta(\mathbf{x}^m) - y^m\right) \frac{\partial h_\theta(\mathbf{x}^m)}{\partial \theta_j} \\ &= \sum_m \left(h_\theta(\mathbf{x}^m) - y^m\right) h_\theta(\mathbf{x}^m) \left(1 - h_\theta(\mathbf{x}^i)\right) \frac{\partial \theta^\mathsf{T} \mathbf{x}^\mathbf{m}}{\partial \theta_j} \\ &= \sum_m \left(h_\theta(\mathbf{x}^m) - y^m\right) h_\theta(\mathbf{x}^m) \left(1 - h_\theta(\mathbf{x}^m)\right) \mathbf{x}_j^m\end{aligned} \tag{29}$$

We set the initial θ as zeros, and start with it. We use the Stochastic Gradient Descend algorithm to minimize the cost function. For the instance $(\mathbf{x}, y)$, the parameters are updated using

$$\begin{aligned}\theta_j{}^{(t+1)} &= \theta_j{}^{(t)} - \alpha \left(\frac{\partial \mathcal{J}(\theta)}{\partial \theta_j} |_{\theta = \theta^{(t)}} \right) \\ &= \theta_j{}^{(t)} - \alpha (h_{\theta^{(t)}}(\mathbf{x}) - y) h_{\theta^{(t)}}(\mathbf{x}) \left(1 - h_{\theta^{(t)}}(\mathbf{x})\right) \mathbf{x}_j\end{aligned} \tag{30}$$

where α is the learning rate, and it is set to 0.01 in our experiments. We summarize the learning method in Algorithm 5. For each combination rule, we iteratively scan

Algorithm 5 Polarity Model Learning Algorithm

Input: $\mathcal{G}_C$: Combination rules, ε: Stopping condition, α: Learning rate
Output: θ: Parameters of the polarity model

```
function EstimatePolarityModel($\mathcal{G}_C$)
    for all combination rule $r \in \mathcal{G}_C$ do
        $\theta^{(0)} \leftarrow (0, 0, ..., 0)^\mathsf{T}$
        repeat
            $\left(\mathbf{x}, y\right) \leftarrow$ randomly select a training instance
            $\theta_j{}^{(t+1)} \leftarrow \theta_j{}^{(t)} - \alpha(h_{\theta^{(t)}}(\mathbf{x}) - y) h_{\theta^{(t)}}(\mathbf{x}) \left(1 - h_{\theta^{(t)}}(\mathbf{x})\right) \mathbf{x}_j$
            $t \leftarrow t + 1$
        until $\left\|\theta^{(t+1)} - \theta^{(t)}\right\|_2^2 < \varepsilon$
        assign $\theta^{(T)}$ as the parameters of the polarity model for rule $r$
```

Algorithm 6 Sentiment Grammar Learning

```
Input: $\mathcal{D}$: Data set $\{(s, \mathcal{L}_s)\}$, $T$: Maximum number of iteration    ▷ $\mathcal{L}_s$: Polarity label of $s$
Output: $\mathcal{G}_\mathcal{D}$: Dictionary rules, $\mathcal{G}_\mathcal{C}$: Combination rules
$\mathcal{G}_\mathcal{C} \leftarrow \{\}$
repeat
    $\mathcal{G}_\mathcal{D} \leftarrow$ MineDictionaryRules($\mathcal{D}, \mathcal{G}_\mathcal{C}$)    ▷ Algorithm 3
    $\mathcal{G}_\mathcal{C} \leftarrow$ MineCombinationRules($\mathcal{D}, \mathcal{G}_\mathcal{D}$)    ▷ Algorithm 4
until iteration number exceeds $T$
EstimatePolarityModel($\mathcal{G}_\mathcal{C}$)    ▷ Algorithm 5
return $\mathcal{G}_\mathcal{D}, \mathcal{G}_\mathcal{C}$
```

through the training examples $(\mathbf{x}, y)$ in a random order, and update the parameters θ according to Equation (30). The stopping condition is $\left\|\theta^{(t+1)} - \theta^{(t)}\right\|_2^2 < \varepsilon$, which indicates the parameters become stable.

4.2.4 Summary of the Grammar Learning Algorithm. We summarize the grammar learning process in Algorithm 6, which learns the sentiment grammar in an iterative manner.

We first learn the dictionary rules and their polarity probabilities by counting the frequencies in negative and positive classes. Only the fragments whose occurrence numbers are larger than the threshold τ_f are kept. As mentioned in Section 4.2.1, the context can essentially change the distribution of sentiment fragments. We take the combination rules into consideration to acquire more reasonable $\mathcal{G}_\mathcal{D}$. In the first iteration, the set of combination rules is empty. Therefore, we have no information about compositionality to improve dictionary rule learning. The initial $\mathcal{G}_\mathcal{D}$ contains some inaccurate sentiment distributions. Next, we replace the subsequences of dictionary rules to their polarity labels, and generalize these sentiment fragments to the combination rules $\mathcal{G}_\mathcal{C}$ as illustrated in Section 4.2.2. At the same time, we can obtain their compositional types and learn parameters of the polarity model. We iterate over these two steps to obtain refined $\mathcal{G}_\mathcal{D}$ and $\mathcal{G}_\mathcal{C}$.

5. Experimental Studies

In this section, we describe experimental results on existing benchmark data sets with extensive comparisons with state-of-the-art sentiment classification methods. We also present the effects of different experimental settings in the proposed statistical sentiment parsing framework.

5.1 Experiment Set-up

We describe the data sets in Section 5.1.1, the experimental settings in Section 5.1.2, and the methods used for comparison in Section 5.1.3.

5.1.1 Data Sets. We conduct experiments on sentiment classification for sentence-level and phrase-level data. The sentence-level data sets contain user reviews and critic reviews from Rotten Tomatoes[3] and IMDb.[4] We balance the positive and negative instances in the training data set to mitigate the problem of data imbalance. Moreover, the Stanford Sentiment Treebank[5] contains polarity labels of all syntactically plausible phrases. In addition, we use the MPQA[6] data set for the phrase-level task. We describe these data sets as follows.

RT-C: 436,000 critic reviews from Rotten Tomatoes. It consists of 218,000 negative and 218,000 positive critic reviews. The average review length is 23.2 words. Critic reviews from Rotten Tomatoes contain a label (Rotten: Negative, Fresh: Positive) to indicate the polarity, which we use directly as the polarity label of corresponding reviews.

PL05-C: The sentence polarity data set v1.0 (Pang and Lee 2005) contains 5,331 positive and 5,331 negative snippets written by critics from Rotten Tomatoes. This data set is widely used as the benchmark data set in the sentence-level polarity classification task. The data source is the same as RT-C.

SST: The Stanford Sentiment Treebank (Socher et al. 2013) is built upon PL05-C. The sentences are parsed to parse trees. Then, 215,154 syntactically plausible phrases are extracted and annotated by workers from Amazon Mechanical Turk. The experimental settings of positive/negative classification for sentences are the same as in Socher et al. (2013).

RT-U: 737,806 user reviews from Rotten Tomatoes. Because we focus on sentence-level sentiment classification, we filter out user reviews that are longer than 200 characters. The average length of these short user reviews from Rotten Tomatoes is 15.4 words. Following previous work on polarity classification, we use the review score to select highly polarized reviews. For the user reviews from Rotten Tomatoes, a negative review has a score <2.5 out of 5, and a positive review has a score >3.5 out of 5.

IMDB-U: 600,000 user reviews from IMDb. The user reviews in IMDb contain comments and short summaries (usually a sentence) to summarize the overall sentiment expressed in the reviews. We use the review summaries as the sentence-level reviews. The average length is 6.6 words. For user reviews of IMDb, a negative review has a score <4 out of 10, and a positive review has a score >7 out of 10.

C-TEST: 2,000 labeled critic reviews sampled from RT-C. We use C-TEST as the testing data set for RT-C. Note that we exclude these from the training data set (i.e., RT-C).

U-TEST: 2,000 manually labeled user reviews sampled from RT-U. User reviews often contain some noisy ratings compared with critic reviews. To eliminate the effect of noise, we sample 2,000 user reviews from RT-U, and annotate their polarity labels manually. We use U-TEST as a testing data set for RT-U and IMDB-U, which are both user reviews. Note that we exclude them from the training data set (i.e., RT-U).

MPQA: The opinion polarity subtask of the MPQA data set (Wiebe, Wilson, and Cardie 2005). The authors manually annotate sentiment polarity labels for the expressions (i.e., sub-sentences) within a sentence. We regard the expressions as short sentences in our experiments. There are 7,308 negative examples and 3,316 positive examples in this data set. The average number of words per example is 3.1.

[3] `http://www.rottentomatoes.com.`

[4] `http://www.imdb.com.`

[5] `http://nlp.stanford.edu/sentiment/treebank.html.`

[6] `http://mpqa.cs.pitt.edu/corpora/mpqa_corpus.`

Table 3
Statistical information of data sets. #Negative and #Positive are the number of negative instances and positive instances, respectively. l_{avg} is average length of sentences in the data set, and $|V|$ is the vocabulary size.

Data Set	Size	#Negative	#Positive	l_{avg}	$\|V\|$
RT-C	436,000	218,000	218,000	23.2	136,006
PL05-C	10,662	5,331	5,331	21.0	20,263
SST	98,796	42,608	56,188	7.5	16,372
RT-U	737,806	368,903	368,903	15.4	138,815
IMDB-U	600,000	300,000	300,000	6.6	83,615
MPQA	10,624	7,308	3,316	3.1	5,992

Table 3 shows the summary of these data sets, and all of them are publicly available at `http://goo.gl/WxTdPf`.

5.1.2 Settings. To compare with other published results for PL05-C and MPQA, the training and testing regime (10-fold cross-validation) is the same as in Pang and Lee (2005), Nakagawa, Inui, and Kurohashi (2010) and Socher et al. (2011). For SST, the regime is the same as in Socher et al. (2013). We use C-TEST as the testing data for RT-C, and U-TEST as the testing data for RT-U and IMDB-U. There are a number of settings that have trade-offs in performance, computation, and the generalization power of our model. The best settings are chosen by a portion of training split data that serves as the validation set. We provide the performance comparisons using different experimental settings in Section 5.4.

Number of training examples: The size of training data has been widely recognized as one of the most important factors in machine learning-based methods. Generally, using more data leads to better performance. By default, all the training data is used in our experiments. We use the same size of training data in different methods for fair comparisons.

Number of training iterations (*T*): We use AdaGrad (Duchi, Hazan, and Singer 2011) as the optimization algorithm in the learning process. The algorithm starts with randomly initialized parameters, and alternates between searching candidate sentiment trees and updating parameters of the ranking model. We treat one-pass scan of training data as an iteration.

Beam size (*K*): The beam size is used to make a trade-off between the search space and the computation cost. Moreover, an appropriate beam size can prune unfavorable candidates. We set $K = 30$ in our experiments.

Regularization (λ): The regularization parameter λ in Equation (22) is used to avoid over-fitting. The value used in the experiments is 0.01.

Minimum fragment frequency: It is difficult to estimate reliable polarity probabilities when the fragment appears very few times. Hence, a minimum fragment frequency that is too small will introduce noise in the fragment learning process. On the other hand, a large threshold will lose much useful information. The minimum fragment frequency is chosen according to the size of the training data set and the validation performance. To be specific, we set this parameter as 4 for RT-C, SST, RT-U, and IMDB-U, and 2 for PL05-C and MPQA.

Maximum fragment length: High order n-grams are more precise and deterministic expressions than unigrams and bigrams. So it would be useful to use long fragments to capture polarity information. According to the experimental results, as the maximum fragment length increases, the accuracy of sentiment classification increases. The maximum fragment length is set to 7 words in our experiments.

5.1.3 Sentiment Classification Methods for Comparison. We evaluate the proposed statistical sentiment parsing framework on the different data sets, and compare the results with some baselines and state-of-the-art sentiment classification methods described as follows.

SVM-m: Support Vector Machine (SVM) achieves good performance in the sentiment classification task (Pang and Lee 2005). Though unigrams and bigrams are reported as the most effective features in existing work (Pang and Lee 2005), we use high-order n-gram ($1 \leq n \leq m$) features to conduct fair comparisons. Hereafter, m has the same meaning. We use LIBLINEAR (Fan et al. 2008) in our experiments because it can handle well the high feature dimension and a large number of training examples. We try different hyper-parameters $C \in \{10^{-2}, 10^{-1}, 1, 5, 10, 20\}$ for SVM, and select C on the validation set.

MNB-m: As indicated in Wang and Manning (2012), Multinomial Naïve Bayes (MNB) often outperforms SVM for sentence-level sentiment classification. We utilize Laplace smoothing (Manning, Raghavan, and Schütze 2008) to tackle the zero probability problem. High order n-gram ($1 \leq n \leq m$) features are considered in the experiments.

LM-m: Language Model (LM) is a generative model calculating the probability of word sequences. It is used for sentiment analysis in Cui, Mittal, and Datar (2006). Probability of generating sentence s is calculated by $P(s) = \prod_{i=0}^{|s|-1} P\left(w_i | w_0^{i-1}\right)$, where w_0^{i-1} denotes the word sequence $w_0 \ldots w_{i-1}$. We use Good-Turing smoothing (Good 1953) to overcome sparsity when estimating the probability of high-order n-gram. We train language models on negative and positive sentences separately. For a sentence, its polarity is determined by comparing the probabilities calculated from the positive and negative language models. The unknown-word token is treated as a regular word (denoted by <*UNK*>). The SRI Language Modeling Toolkit (Stolcke 2002) is used in our experiment.

Voting-w/Rev: This approach is proposed by Choi and Cardie (2009b), and is used as a baseline in Nakagawa, Inui, and Kurohashi (2010). The polarity of a subjective sentence is decided by the voting of each phrase's prior polarity. The polarity of phrases that have odd numbers of negation phrases in their ancestors is reversed. The results are reported by Nakagawa, Inui, and Kurohashi (2010).

HardRule: This baseline method is compared by Nakagawa, Inui, and Kurohashi (2010). The polarity of a subjective sentence is deterministically decided based on rules, by considering the sentiment polarity of dependency subtrees. The polarity of a modifier is reversed if its head phrase has a negation word. The decision rules are applied from the leaf nodes to the root node in a dependency tree. We use the results reported by Nakagawa, Inui, and Kurohashi (2010).

Tree-CRF: Nakagawa, Inui, and Kurohashi (2010) present a dependency tree-based method using conditional random fields with hidden variables. In this model, the polarity of each dependency subtree is represented by a hidden variable. The value of the hidden variable of the root node is identified as the polarity of the whole sentence. The experimental results are reported by Nakagawa, Inui, and Kurohashi (2010).

RAE-pretrain: Socher et al. (2011) introduce a framework based on recursive autoencoders to learn vector space representations for multi-word phrases and predict sentiment distributions for sentences. We use the results with pre-trained word vectors learned on Wikipedia, which leads to better results compared with randomized word vectors. We directly compare the results with those in Socher et al. (2011).

MV-RNN: Socher et al. (2012) try to capture the compositional meaning of long phrases through matrix-vector recursive neural networks. This model assigns a vector and a matrix to every node in the parse tree. Matrices are regarded as operators, and vectors capture the meaning of phrases. The results are reported by Socher et al. (2012, 2013).

s.parser-LongMatch: The longest matching rules are utilized in the decoding process. In other words, the derivations that contain the fewest rules are used for all text spans. In addition, the dictionary rules are preferred to the combination rules if both of them match the same text span. The dynamic programming algorithm is used in the implementation.

s.parser-w/oComb: This is our method without using the combination rules (such as $N \rightarrow$ not P) learned from data.

5.2 Results of Sentiment Classification

We present the experimental results of the sentiment classification methods on the different data sets in Table 4. The top three methods on each data set are in bold, and the best methods are also underlined. The experimental results show that s.parser achieves better performance than other methods on most data sets.

The data sets RT-C, PL05-C, and SST are critic reviews. On RT-C, the accuracy of s.parser increases by 2 percentage points, 2.9 percentage points, and 7.1 percentage points from the best results of SVM, MNB, and LM, respectively. On PL05-C, the accuracy of s.parser also rises by 2.1 percentage points, 0.7 percentage points, and 4.4 percentage points from the best results of SVM, MNB, and LM, respectively. Compared to Voting-w/Rev and HardRule, s.parser outperforms them by 16.4 percentage points and 16.6 percentage points. The results indicate that our method significantly outperforms the baselines that use manual rules, as rule-based methods lack a probabilistic way to model the compositionality of context. Furthermore, s.parser achieves an accuracy improvement rate of 2.2 percentage points, 1.8 percentage points, and 0.5 percentage points over Tree-CRF, RAE-pretrain, and MV-RNN, respectively. On SST, s.parser outperforms SVM, MNB, and LM by 3.4 percentage points, 1.4 percentage points, and 3.8 percentage points, respectively. The performance is better than MV-RNN with an improvement rate of 1.8 percentage points. Moreover, the result is comparable to the 85.4% obtained by recursive neural tensor networks (Socher et al. 2013) without depending on syntactic parsing results.

On the user review data sets RT-U and IMDB-U, our method also achieves the best results. More specifically, on the data set RT-U, s.parser outperforms the best results of SVM, MNB, and LM by 1.7 percentage points, 2.9 percentage points, and 1.5 percentage points, respectively. On the data set IMDB-U, our method brings an improved accuracy rate of 2.1 percentage points, 3.7 percentage points, and 2.2 percentage points over SVM, MNB, and LM, respectively. We find that MNB performs better than SVM and LM on the critics review data sets RT-C and PL05-C. Also, SVM and LM achieve better results on the user review data sets RT-U and IMDB-U. The s.parser is more robust for the different genres of data sets.

Table 4
Sentiment classification results on different data sets; The top three methods are in **bold** and the best is also **underlined**; SVM-m = Support Vector Machine; MNB-m = Multinomial Naïve Bayes; LM-m = Language Model; Voting-w/Rev = Voting with negation rules; HardRule = Rule based method on dependency tree; Tree-CRF = Dependency tree-based method employing conditional random fields; RAE-pretrain = Recursive autoencoders with pre-trained word vectors; MV-RNN = Matrix-vector recursive neural network; s.parser-LongMatch = The longest matching rules are used; s.parser-w/oComb = Without using the combination rules; s.parser = Our method. Some of the results are missing (indicated by "-") in the table as there is no publicly available implementation or they are hard to scale up.

Method	RT-C	PL05-C	SST	RT-U	IMDB-U	MPQA
SVM-1	80.3	76.3	81.1	88.5	84.9	85.1
SVM-2	**83.0**	77.4	81.3	88.9	86.8	85.3
SVM-3	**83.1**	77.0	81.2	89.7	**87.2**	85.5
SVM-4	81.5	76.9	80.9	**89.8**	87.0	85.6
SVM-5	81.7	76.8	80.8	89.3	87.0	85.6
MNB-1	79.6	78.0	82.6	83.3	82.7	85.0
MNB-2	82.0	**78.8**	**83.3**	87.5	85.6	85.0
MNB-3	82.2	78.4	**82.9**	88.6	84.6	85.0
MNB-4	81.8	78.2	82.6	88.2	83.1	85.1
MNB-5	81.7	78.1	82.4	88.1	82.5	85.1
LM-1	77.6	75.1	80.9	87.6	81.8	64.0
LM-2	78.0	74.1	78.4	89.0	85.8	71.4
LM-3	77.3	74.2	78.3	89.3	**87.1**	71.1
LM-4	77.2	73.0	78.3	89.6	87.0	71.1
LM-5	77.0	72.9	78.2	**90.0**	**87.1**	71.1
Voting-w/Rev	-	63.1	-	-	-	81.7
HardRule	-	62.9	-	-	-	81.8
Tree-CRF	-	77.3	-	-	-	**86.1**
RAE-pretrain	-	77.7	-	-	-	**86.4**
MV-RNN	-	**79.0**	**82.9**	-	-	-
s.parser-LongMatch	82.8	78.6	82.5	89.4	86.9	85.7
s.parser-w/oComb	82.6	78.3	82.4	89.0	86.4	85.5
s.parser	**85.1**	**79.5**	**84.7**	**91.5**	**89.3**	**86.2**

On the data set MPQA, the accuracy of s.parser increases by 0.5 percentage points, 1.1 percentage points, and 14.8 percentage points from the best results of SVM, MNB, and LM, respectively. Compared with Voting-w/Rev and HardRule, s.parser achieves 4.5 percentage point and 4.4 percentage point improvements over them. As illustrated in Table 3, the size and length of sentences in MPQA are much smaller than those in the other four data sets. The RAE-pretrain achieves better results than other methods on this data set, because the word embeddings pre-trained on Wikipedia can leverage smoothing to relieve the sparsity problem in MPQA. If we do not use any external resources (i.e., Wikipedia), the accuracy of RAE on MPQA is 85.7%, which is lower than Tree-CRF and s.parser. The results indicate that s.parser achieves the best result if no external resource is used.

In addition, we compare the results of s.parser-LongMatch and s.parser-w/oComb. The s.parser-LongMatch utilizes the dictionary rules and combination rules in the longest matching manner, whereas s.parser-w/oComb removes the combination rules in the parsing process. Compared with the results of s.parser, we find that both the ranking model and the combination rules play a positive role in the model. The ranking model learns to score parse trees by assigning larger weights to the rules that tend to obtain correct labels. Also, the combination rules generalize these dictionary rules to deal with the sentiment compositionality in a symbolic way, which enables the model to process unseen phrases. Furthermore, s.parser-LongMatch achieves better results than s.parser-w/oComb. This indicates that the effects of the combination rules are more pronounced than the ranking model.

The bag-of-words classifiers work well for long documents relying on sentiment words that appear many times in a document. The redundancy characteristics provide strong evidence for sentiment classification. Even though some phrases of a document are not estimated accurately, it can still result in a correct polarity label. However, for short text, such as a sentence, the compositionality plays an important role in sentiment classification. Tree-CRF, MV-RNN, and s.parser take compositionality into consideration in different ways, and they achieve significant improvements over SVM, MNB, and LM. We also find that the high order *n*-grams contribute to classification accuracy on most of the data sets, but they harm the accuracy of LM on PL05-C. The high-order *n*-grams can partially solve compositionality in a brute-force way.

5.3 Effect of Training Data Size

We further investigate the effect of the size of training data for different sentiment classification methods. This is meaningful as the number of the publicly available reviews is increasing dramatically nowadays. The methods that can take advantage of more training data will be even more useful in practice.

We report the results of s.parser compared with SVM, MNB, and LM on the data set RT-C using different training data size. In order to make the figure clear, we only present the results of SVM/MNB/LM-1/5 here. As shown in Figure 9, we find that the size of training data plays an important role for all these sentiment classification methods. The basic conclusion is that the performance of all the methods rise as the data size increases, especially when the data size is smaller than a certain number. It meets our intuition that the size of data is the key factor when the size is relatively small. When the size of data is larger, the growth of accuracy becomes slower. The performance of the baseline methods starts to converge after the data size is larger than 200,000. The comparisons illustrate that s.parser significantly outperforms these baselines. And the performance of s.parser becomes even better when the data size increases. The convergence of s.parser's performance is slower than the others. It indicates that s.parser leverages data more effectively and benefits more from a larger data set. With more training data, s.parser learns more dictionary rules and combination rules. These rules enhance the generalization ability of our model. Furthermore, it estimates more reliable parameters for the polarity model and ranking model. In contrast, the bag-of-words based approaches (such as SVM, MNB, and LM) cannot make full use of high-order information in the data set. The generalization ability of the combination rules of s.parser leads to better performance, and take advantage of larger data. It should be noted that there are similar trends with the other data sets.

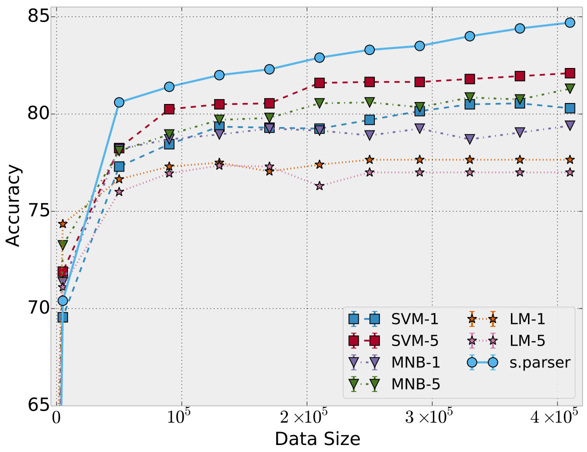
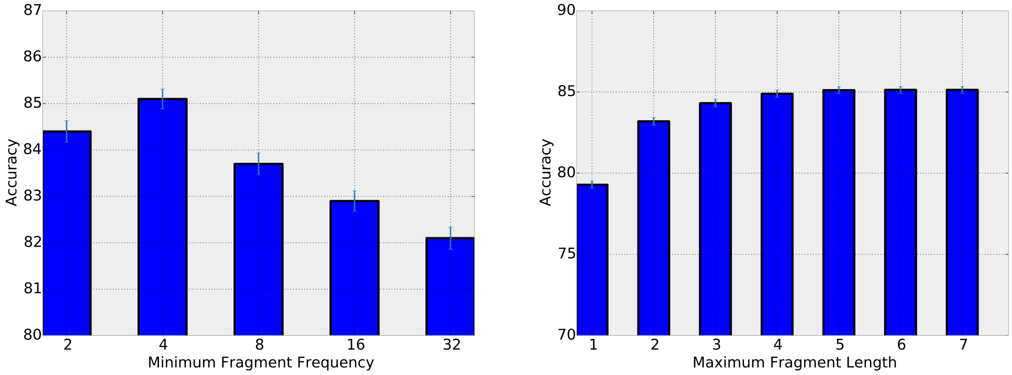


Figure 9
The curves show the test accuracy as the number of training examples increases. Our method s.parser significantly outperforms the other methods, which indicates s.parser can leverage data more effectively and benefits more from larger data.

5.4 Effect of Experimental Settings

In this section, we investigate the effects of different experimental settings. We show the results on the data set RT-C by only changing one factor and fixing the others.

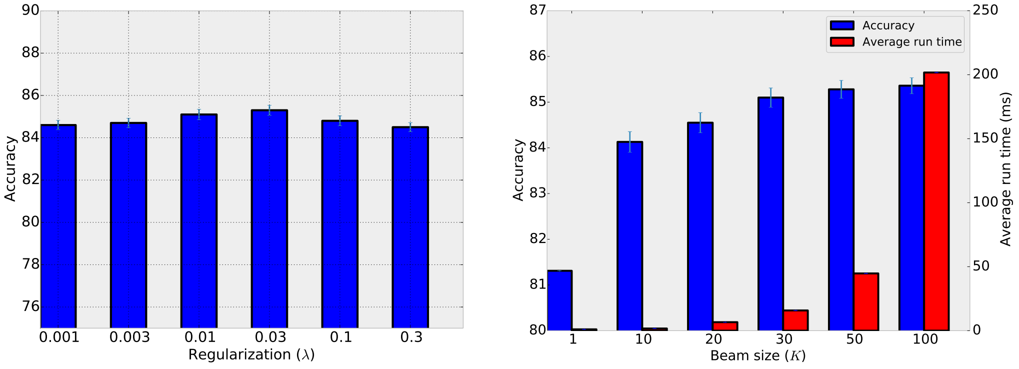
Figure 10 shows the effect of minimum fragment frequency, and maximum fragment length. Specifically, Figure 10a indicates that a minimum fragment frequency that is too small will introduce noise, and it is difficult to estimate reliable polarity probabilities for infrequent fragments. However, a minimum fragment frequency that is too large will discard too much useful information. As shown in Figure 10b, we find that accuracy increases as the maximum fragment length increases. The results illustrate that the large maximum fragment length is helpful for s.parser. We can learn more



(a) Effect of minimum fragment frequency

(b) Effect of maximum fragment length

Figure 10
(a) When the minimum fragment frequency is small, noise is introduced in the fragment learning process. On the other hand, too large a threshold loses useful information. (b) As the maximum fragment length increases, the accuracy increases monotonically. It indicates that long fragments are useful for our method.



(a) Effect of regularization

(b) Effect of beam size

Figure 11
(a) The test accuracy is relatively insensitive to the regularization parameter λ in Equation (22).
(b) As the beam size K increases, the test accuracy increases; however, the computation costs also become more expensive. When $K = 1$, the optimization algorithm cannot learn any weights.

combination rules with a larger maximum fragment length, and long dictionary rules capture more precise expressions than unigrams. This conclusion is the same as that in Section 5.2.

As shown in Figure 11, we also investigate how the training iteration, regularization, and beam size affect the results. As shown in Figure 11a, we try a wide range of regularization parameters λ in Equation (22). The results indicate that it is insensitive to the choice of λ. Figure 11b shows the effects of different beam size K in the search process. When beam size $K = 1$, the optimization algorithm cannot learn the weights. In this case, the decoding process is to select one search path randomly, and compute its polarity probabilities. The results become better as the beam size K increases. On the other hand, the computation costs increase. The proper beam size K can prune some candidates to speed up the search procedure. It should be noted that the sentence length also effects the run time.

5.5 Results of Grammar Learning

The sentiment grammar plays a central role in the statistical sentiment parsing framework. It is obvious that the accuracy of s.parser relies on the quality of the automatically learned sentiment grammar. The quality can be implicitly evaluated by the accuracy of sentiment classification results, as we have shown in previous sections. However, there is no straightforward way to explicitly evaluate the quality of the learned grammar. In this section, we provide several case studies of the learned dictionary rules and combination rules to further illustrate the results of the sentiment grammar learning process as detailed in Section 4.2.

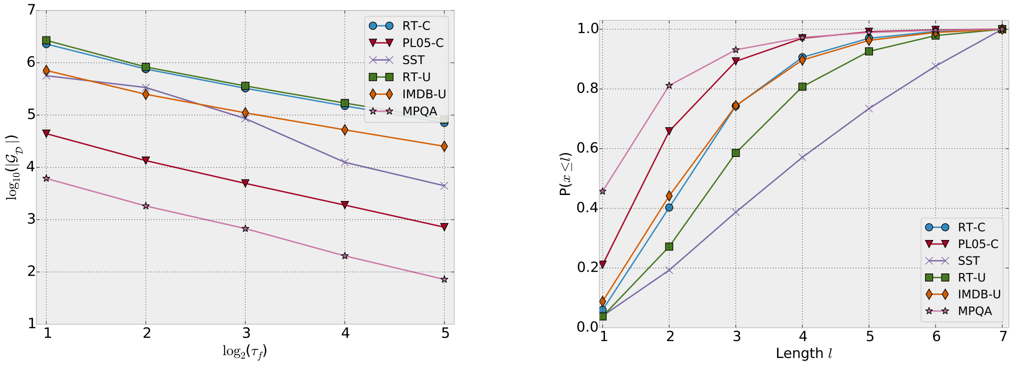
To start with, we report the total number of dictionary rules and combination rules learned from the data sets. As shown in Table 5, the results indicate that we can learn more dictionary rules and combination rules from the larger data sets. Although we learn more dictionary rules from RT-C than from IMDB-U, the number of combination rules learned from RT-C is less than from IMDB-U. It indicates that the language usage of RT-C is more diverse than that of IMDB-U. For SST, more rules are learned because of its constituent-level annotations.

Table 5
Number of rules learned from different data sets. τ_f represents minimum fragment frequency, $|\mathcal{G}_\mathcal{D}|$ represents total number of dictionary rules, and $|\mathcal{G}_\mathcal{C}|$ is the total number of combination rules.

Data Set	τ_f	$\lvert\mathcal{G}_\mathcal{D}\rvert$	$\lvert\mathcal{G}_\mathcal{C}\rvert$
RT-C	4	758,723	952
PL05-C	2	44,101	139
SST	4	336,695	751
RT-U	4	831,893	2,003
IMDB-U	4	249,718	1,014
MPQA	2	6,146	21

Furthermore, we explore how the minimum fragment frequency τ_f affects the number of dictionary rules, and present the distribution of dictionary rule length. As illustrated in Figure 12a, we find that the relation between total number of dictionary rules $|\mathcal{G}_\mathcal{D}|$ and minimum fragment frequency τ_f obeys the power law, that is, the $\log_{10}(|\mathcal{G}_\mathcal{D}|) - \log_2(\tau_f)$ graph takes a linear form. It indicates that most of the fragments appear few times, and only some of them appear frequently. Notably, all the syntactically plausible phrases of SST are annotated, so its distribution is different from the other sentence-level data sets. Figure 12b shows the cumulative distribution of dictionary rule length l. It presents most dictionary rules as short ones. For all data sets except SST, more than 80% of dictionary rules are shorter than five words. The length distributions of data sets RT-C and IMDB-U are similar, whereas we obtain more high order n-grams from RT-U and SST.

We further investigate the effect of context for dictionary rule learning. Table 6 shows some dictionary rules with polarity probabilities learned by our method and



(a) Effect of minimum fragment frequency $\log_2(\tau_f)$

(b) Cumulative distribution of dictionary rule length l

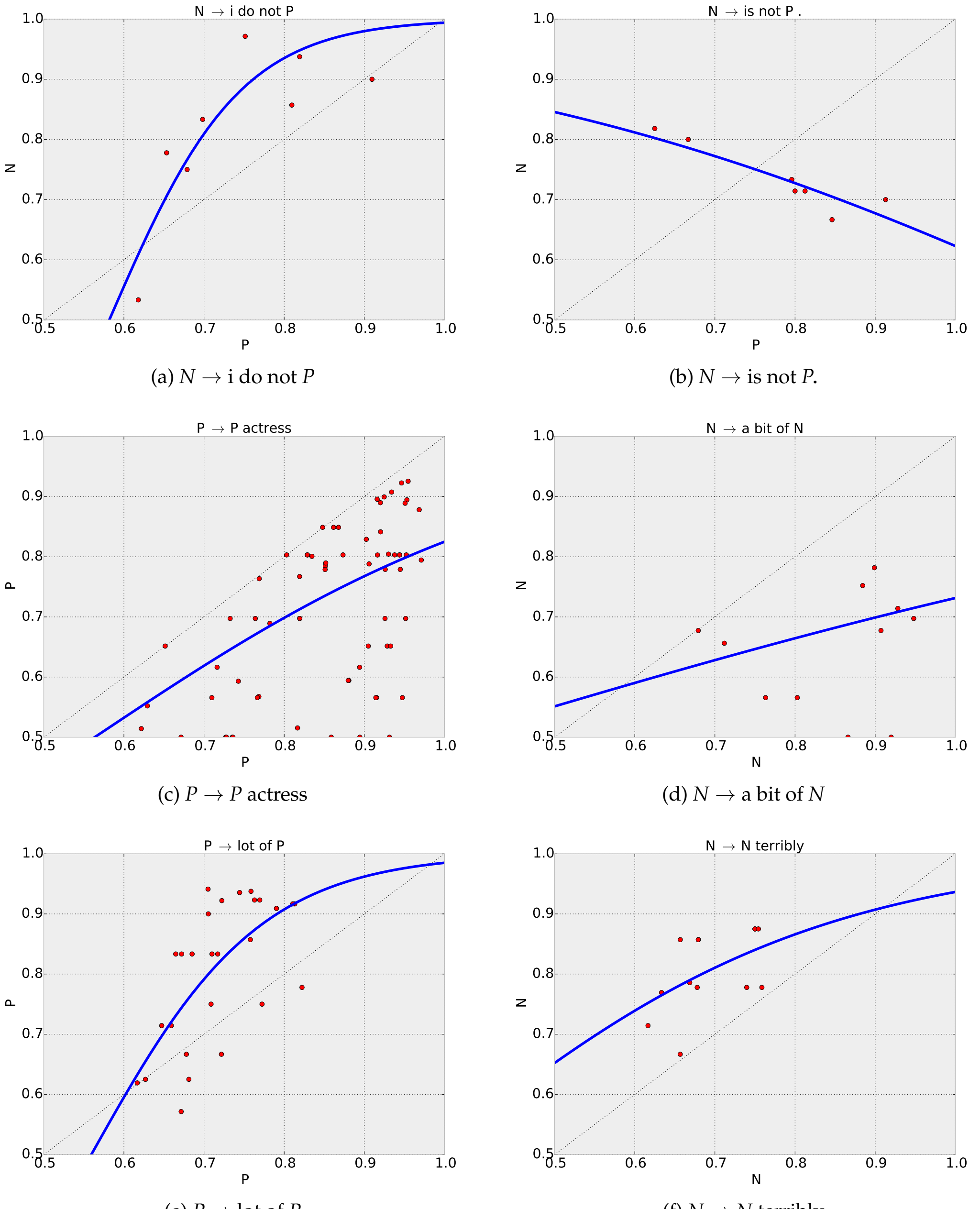
Figure 12
(a) We choose $\tau_f = 2, 4, 8, 16, 32$, and plot $\log_{10}(|\mathcal{G}_\mathcal{D}|)$–$\log_2(\tau_f)$ graph to show the effects of τ_f for total number of dictionary rules $|\mathcal{G}_\mathcal{D}|$. The results (except SST) follow a power law distribution. (b) The cumulative distribution of dictionary rule length l indicates that most dictionary rules are short ones.

Table 6
Comparing our dictionary rule learning method with naive counting. The dictionary rules that are assigned different polarities by these two methods are presented. $\mathcal{N}$ represents negative, and $\mathcal{P}$ represents positive. The polarity probabilities of fragments are shown in this table, and they demonstrate that our method learns more intuitive results than counting directly.

Fragment	Naive Count			s.parser		
	$\mathcal{N}$	$\mathcal{P}$	Polarity	$\mathcal{N}$	$\mathcal{P}$	Polarity
are fun	0.54	0.46	$\mathcal{N}$	0.11	0.89	$\mathcal{P}$
a very good movie	0.61	0.39	$\mathcal{N}$	0.19	0.81	$\mathcal{P}$
looks gorgeous	0.56	0.44	$\mathcal{N}$	0.17	0.83	$\mathcal{P}$
to enjoy the movies	0.53	0.47	$\mathcal{N}$	0.14	0.86	$\mathcal{P}$
is corny	0.43	0.57	$\mathcal{P}$	0.83	0.17	$\mathcal{N}$
′ s flawed	0.32	0.68	$\mathcal{P}$	0.63	0.37	$\mathcal{N}$
a difficult film to	0.43	0.57	$\mathcal{P}$	0.67	0.33	$\mathcal{N}$
disappoint	0.39	0.61	$\mathcal{P}$	0.77	0.23	$\mathcal{N}$

naive counting on RT-C. We notice that if we count the fragment occurrence number directly, some polarities of fragments are learned incorrectly. This is caused by the effect of context as described in Section 4.2.1. By taking the context into consideration, we obtain more reasonable polarity probabilities of dictionary rules. Our dictionary rule learning method takes compositionality into consideration, namely, we skip the count if there exist some negation indicators outside the phrase. This constraint tries to ensure that the polarity of fragments is the same as the whole sentence. As shown in the results, the polarity probabilities learned by our method are more reasonable and meet people's intuitions. However, there are also some negative examples caused by "false subjective." For instance, the neutral phrase *to pay it* tends to appear in negative sentences, and it is learned as a negative phrase. This makes sense for the data distribution, but it may lead to the mismatch for the combination rules.

In Figure 13, we show the polarity model of some combination rules learned from the data set RT-C. The first two examples are negation rules. We find that both switch negation and shift negation exist in data, instead of using only one negation type in previous work (Choi and Cardie 2008; Saurí 2008; Taboada et al. 2011). For the rule "$N \rightarrow$ i do not P," we find that it is a switch negation rule. This rule reverses the polarity and the corresponding polarity strength. For instance, the *i do not like it very much* is more negative than the *i do not like it*. As shown in Figure 13b, the "$N \rightarrow$ is not P" is a shift negation that reduces a fixed polarity strength to reverse the original polarity. Specifically, the *is not good* is more negative than the *is not great*, as described in Section 3.4. We have a similar conclusion for the next two weaken rules. As illustrated in Figure 13c, the "$P \rightarrow P$ actress" describes one aspect of a movie, hence it is more likely to decrease the polarity intensity. We find that this rule is a fixed intensification rule that reduces the polarity probability by a fixed value. The "$N \rightarrow$ a bit of N" is a percentage intensification rule, which scales polarity intensity by a percentage. It reduces more strength for stronger polarity. The last two rules in Figure 13e and Figure 13f are strengthen rules. Both "$P \rightarrow$ lot of P" and "$N \rightarrow N$ terribly" increase the polarity strength of the sub-fragments. These cases indicate that it is necessary to learn how the context performs compositionality from data. In order to capture the compositionality for different rules, we define the polarity model and learn parameters for each rule. This also agrees with the models of Socher et al. (2012) and Dong et al. (2014),



(a) $N \rightarrow$ i do not P

(b) $N \rightarrow$ is not P.

(c) $P \rightarrow P$ actress

(d) $N \rightarrow$ a bit of N

(e) $P \rightarrow$ lot of P

(f) $N \rightarrow N$ terribly

Figure 13
Illustration of the polarity model for combination rules: (a)(b) Negation rule. (c)(d) Weaken rule. (e)(f) Strengthen rule. The labels of axes represent the corresponding polarity labels, the red points are the training instances, and the blue lines are the regression results for the polarity model.

which use multiple composition matrices to make compositions specific and is an improvement over the recursive neural network that employs one composition matrix.

6. Conclusion and Future Work

In this article, we propose a statistical parsing framework for sentence-level sentiment classification that provides a novel approach to designing sentiment classifiers from a new perspective. It directly analyzes the sentiment structure of a sentence other than relying on syntactic parsing results, as in existing literature. We show that complicated phenomena in sentiment analysis, such as negation, intensification, and contrast, can be handled in a similar manner to simple and straightforward sentiment expressions in a unified and probabilistic way. We provide a formal model to represent the sentiment grammar built upon Context-Free Grammars. The framework consists of: (1) a parsing model to analyze the sentiment structure of a sentence; (2) a polarity model to calculate sentiment strength and polarity for each text span in the parsing process; and (3) a ranking model to select the best parsing result from a list of candidate sentiment parse trees. We show that the sentiment parser can be trained from the examples of sentences annotated only with sentiment polarity labels but without using any syntactic or sentiment annotations within sentences. We evaluate the proposed framework on standard sentiment classification data sets. The experimental results show that the statistical sentiment parsing notably outperforms the baseline sentiment classification approaches.

We believe the work on statistical sentiment parsing can be advanced from many different perspectives. First, statistical parsing has been a well-established research field, in which many different grammars and parsing algorithms have been proposed in previously published literature. It will be an interesting direction to apply and adjust more advanced models and algorithms from the syntactic parsing and the semantic parsing to our framework. We leave it as a line of future work. Second, we can incorporate target and aspect information in the statistical sentiment parsing framework to facilitate the target-dependent and aspect-based sentiment analysis. Intuitively, this can be done by introducing semantic tags of targets and aspects as new non-terminals in the sentiment grammar and revising grammar rules accordingly. However, acquiring training data will be an even more challenging task, as we need more fine-grained information. Third, as the statistical sentiment parsing produces more fine-grained information (e.g., the basic sentiment expressions from the dictionary rules as well as the sentiment structure trees), we will have more opportunities to generate better opinion summaries. Moreover, we are interested in jointly learning parameters of the polarity model and the parsing model from data. Last but not the least, we are interested in investigating the domain adaptation, which is a very important and challenging problem in sentiment analysis. Generally, we may need to learn domain-specific dictionary rules for different domains, whereas we believe combination rules are mostly generic across different domains. This is also worth consideration for further study.

Acknowledgments

This research was partly supported by NSFC (grant no. 61421003) and the fund of the State Key Lab of Software Development Environment (grant no. SKLSDE-2015ZX-05). We gratefully acknowledge helpful discussions with Dr. Nan Yang and the anonymous reviewers.

References

Agarwal, Apoorv, Boyi Xie, Ilia Vovsha, Owen Rambow, and Rebecca Passonneau. 2011. Sentiment analysis of twitter data. In *Proceedings of the Workshop on Languages in Social Media*, LSM '11, pages 30–38, Stroudsburg, PA.

Agrawal, Rakesh and Ramakrishnan Srikant. 1994. Fast algorithms for mining association rules in large databases. In *Proceedings of the 20th International Conference on Very Large Data Bases*, VLDB '94, pages 487–499, San Francisco, CA.

Artzi, Yoav and Luke Zettlemoyer. 2013. Weakly supervised learning of semantic parsers for mapping instructions to actions. *Transactions of the Association for Computational Linguistics*, 1(1):49–62.

Bao, Junwei, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 967–976, Baltimore, MD.

Charniak, Eugene. 1997. Statistical parsing with a context-free grammar and word statistics. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Conference on Innovative Applications of Artificial Intelligence*, AAAI'97/IAAI'97, pages 598–603, Providence, RI.

Charniak, Eugene and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 173–180, Stroudsburg, PA.

Chiang, David. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Choi, Yejin and Claire Cardie. 2008. Learning with compositional semantics as structural inference for subsentential sentiment analysis. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '08, pages 793–801, Stroudsburg, PA.

Choi, Yejin and Claire Cardie. 2009a. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 – Volume 2*, EMNLP '09, pages 590–598, Stroudsburg, PA.

Choi, Yejin and Claire Cardie. 2009b. Adapting a polarity lexicon using integer linear programming for domain-specific sentiment classification. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2 – Volume 2*, pages 590–598, Singapore.

Choi, Yejin and Claire Cardie. 2010. Hierarchical sequential learning for extracting opinions and their attributes. In *Proceedings of the ACL 2010 Conference Short Papers*, ACLShort '10, pages 269–274, Stroudsburg, PA.

Chomsky, Noam. 1956. Three models for the description of language. *IRE Transactions on Information Theory*, 2(3):113–124.

Clarke, James, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the world's response. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, CoNLL '10, pages 18–27, Stroudsburg, PA.

Cocke, John. 1969. *Programming Languages and Their Compilers: Preliminary Notes*. Courant Institute of Mathematical Sciences, New York University.

Councill, Isaac G., Ryan McDonald, and Leonid Velikovich. 2010. What's great and what's not: Learning to classify the scope of negation for improved sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, NeSp-NLP '10, pages 51–59, Stroudsburg, PA.

Cui, Hang, Vibhu Mittal, and Mayur Datar. 2006. Comparative experiments on sentiment classification for online product reviews. In *Proceedings of the 21st National Conference on Artificial Intelligence - Volume 2*, AAAI'06, pages 1,265–1,270, Boston, MA.

Davidov, Dmitry, Oren Tsur, and Ari Rappoport. 2010. Enhanced sentiment learning using twitter hashtags and smileys. In *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*, COLING '10, pages 241–249, Stroudsburg, PA.

de Marneffe, Marie-Catherine, Christopher D. Manning, and Christopher Potts. 2010. "was it good? it was provocative." Learning the meaning of scalar adjectives. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 167–176, Stroudsburg, PA.

Dong, Li, Furu Wei, Ming Zhou, and Ke Xu. 2014. Adaptive multi-compositionality for recursive neural models with applications to sentiment analysis. In *AAAI Conference on Artificial Intelligence*, pages 1,537–1,543, Quebec.

Duchi, John, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Fan, Rong-En, Kai-Wei Chang, Cho-Jui Hsieh, Xiang-Rui Wang, and Chih-Jen Lin. 2008. Liblinear: A library for large linear classification. *Journal of Machine Learning Research*, 9:1871–1874.

Good, Irving John. 1953. The population frequencies of species and the estimation of population parameters. *Biometrika*, 40(3-4):237–264.

Goodman, Joshua. 1999. Semiring parsing. *Computational Linguistics*, 25(4):573–605.

Hall, David, Greg Durrett, and Dan Klein. 2014. Less grammar, more features. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 228–237, Baltimore, MD.

Hatzivassiloglou, Vasileios and Kathleen R. McKeown. 1997. Predicting the semantic orientation of adjectives. In *Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, ACL '98, pages 174–181, Stroudsburg, PA.

Jia, Lifeng, Clement Yu, and Weiyi Meng. 2009. The effect of negation on sentiment analysis and retrieval effectiveness. In *Conference on Information and Knowledge Management*, pages 1,827–1,830, Hong Kong.

Kaji, Nobuhiro and Masaru Kitsuregawa. 2007. Building lexicon for sentiment analysis from massive collection of HTML documents. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1,075–1,083, Prague.

Kamps, Jaap, Robert J. Mokken, Maarten Marx, and Maarten de Rijke. 2004. Using WordNet to measure semantic orientation of adjectives. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, volume IV, pages 1,115–1,118, Paris.

Kanayama, Hiroshi and Tetsuya Nasukawa. 2006. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 355–363, Stroudsburg, PA.

Kasami, Tadao. 1965. An efficient recognition and syntax-analysis algorithm for context-free languages. Technical report AFCRL-65-758, Air Force Cambridge Research Lab, Bedford, MA.

Kate, Rohit J. and Raymond J. Mooney. 2006. Using string-kernels for learning semantic parsers. In *ACL 2006: Proceedings of the 21st International Conference on Computational Linguistics and the 44th Annual Meeting of the ACL*, pages 913–920, Morristown, NJ.

Kennedy, Alistair and Diana Inkpen. 2006. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22:110–125.

Klein, Dan and Christopher D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 423–430, Stroudsburg, PA.

Klenner, Manfred, Stefanos Petrakis, and Angela Fahrni. 2009. Robust compositional polarity classification. In *Proceedings of the International Conference RANLP-2009*, pages 180–184, Borovets.

Krestel, Ralf and Stefan Siersdorfer. 2013. Generating contextualized sentiment lexica based on latent topics and user ratings. In *Proceedings of the 24th ACM Conference on Hypertext and Social Media*, HT '13, pages 129–138, New York, NY.

Krishnamurthy, Jayant and Tom M. Mitchell. 2012. Weakly supervised training of semantic parsers. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 754–765, Stroudsburg, PA.

Kübler, Sandra, Ryan McDonald, and Joakim Nivre. 2009. Dependency parsing. *Synthesis Lectures on Human Language Technologies*, 1(1):1–127.

Li, Peng, Yang Liu, and Maosong Sun. 2013. An extended ghkm algorithm for inducing Lambda-SCFG. In *Conference on Artificial Intelligence*, pages 605–611, Bellevue, WA.

Liang, Percy, Michael I. Jordan, and Dan Klein. 2013. Learning dependency-based compositional semantics. *Computational Linguistics*, 39(2):389–446.

Liu, Bing. 2012. *Sentiment Analysis and Opinion Mining*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Liu, Jingjing and Stephanie Seneff. 2009. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, pages 161–169, Stroudsburg, PA.

Liu, Shizhu, Gady Agam, and David A. Grossman. 2012. Generalized sentiment-bearing expression features for sentiment analysis. In *International Conference on Computational Linguistics*, pages 733–744, Mumbai.

Lu, Yue, Malú Castellanos, Umeshwar Dayal, and ChengXiang Zhai. 2011. Automatic construction of a context-aware sentiment lexicon: an optimization approach. In *International World Wide Web Conference*, pages 347–356, Hyderabad.

Maas, Andrew L., Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*, HLT '11, pages 142–150, Stroudsburg, PA.

Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval*. Cambridge University Press, New York.

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

Matsumoto, Shotaro, Hiroya Takamura, and Manabu Okumura. 2005. Sentiment classification using word sub-sequences and dependency sub-trees. In *Proceedings of the 9th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining*, PAKDD'05, pages 301–311, Hanoi.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 91–98, Stroudsburg, PA.

McDonald, Ryan, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. 2007. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the Association for Computational Linguistics (ACL)*, pages 432–439, Prague.

Moilanen, Karo and Stephen Pulman. 2007. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing (RANLP 2007)*, pages 378–382, Borovets.

Moilanen, Karo, Stephen Pulman, and Yue Zhang. 2010. Packed feelings and ordered sentiments: Sentiment parsing with quasi-compositional polarity sequencing and compression. In *Proceedings of the 1st Workshop on Computational Approaches to Subjectivity and Sentiment Analysis (WASSA 2010) at the 19th European Conference on Artificial Intelligence (ECAI 2010)*, pages 36–43, Lisbon.

Mudinas, Andrius, Dell Zhang, and Mark Levene. 2012. Combining lexicon and learning based approaches for concept-level sentiment analysis. In *Proceedings of the First International Workshop on Issues of Sentiment Discovery and Opinion Mining*, WISDOM '12, pages 5:1–5:8, New York, NY.

Nakagawa, Tetsuji, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using CRFS with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 786–794, Stroudsburg, PA.

Pang, Bo and Lillian Lee. 2004. A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics (ACL'04), Main Volume*, pages 271–278, Barcelona.

Pang, Bo and Lillian Lee. 2005. Seeing stars: exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 115–124, Stroudsburg, PA.

Pang, Bo and Lillian Lee. 2008. Opinion mining and sentiment analysis. *Foundations and Trends in Information Retrieval*, 2(1-2):1–135.

Pang, Bo, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up?: Sentiment classification using machine learning techniques. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, EMNLP '02, pages 79–86, Stroudsburg, PA.

Polanyi, Livia and Annie Zaenen. 2006. Contextual valence shifters. In J. G. Shanahan, Y. Qu, and J. Wiebe, editors, *Computing Attitude and Affect in Text: Theory and Applications*. Springer Netherlands, pages 1–10.

Quirk, R. 1985. *A Comprehensive Grammar of the English Language*. Longman.

Raymond, Ruifang Ge and J. Mooney. 2006. Discriminative reranking for semantic parsing. In *Proceedings of the COLING/ACL on Main Conference Poster Sessions*, COLING-ACL '06, pages 263–270, Stroudsburg, PA.

Robbins, H. and S. Monro. 1951. A stochastic approximation method. *Annals of Mathematical Statistics*, 22:400–407.

Saurí, Roser. 2008. *A Factuality Profiler for Eventualities in Text*. Ph.D. thesis, Brandeis University.

Shieber, Stuart M., Yves Schabes, and Fernando C. N. Pereira. 1995. Principles and implementation of deductive parsing. *Journal of Logic Programming*, 24(1–2):3–36.

Socher, Richard, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic compositionality through recursive matrix-vector spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 1,201–1,211, Stroudsburg, PA.

Socher, Richard, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 151–161, Stroudsburg, PA.

Socher, Richard, Alex Perelygin, Jean Y. Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1,631–1,642, Seattle, WA.

Stolcke, Andreas. 2002. SRILM: An extensible language modeling toolkit. In *Proceedings of the 7th International Conference on Spoken Language Processing (ICSLP 2002*, pages 901–904, Denver, CO.

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

Täckström, Oscar and Ryan McDonald. 2011a. Discovering fine-grained sentiment with latent variable structured prediction models. In *Proceedings of the 33rd European Conference on Advances in Information Retrieval*, ECIR'11, pages 368–374, Berlin.

Täckström, Oscar and Ryan McDonald. 2011b. Semi-supervised latent variable models for sentence-level sentiment analysis. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: Short Papers - Volume 2*, HLT '11, pages 569–574, Stroudsburg, PA.

Takamura, Hiroya, Takashi Inui, and Manabu Okumura. 2005. Extracting semantic orientations of words using spin model. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL '05, pages 133–140, Stroudsburg, PA.

Tu, Zhaopeng, Yifan He, Jennifer Foster, Josef van Genabith, Qun Liu, and Shouxun Lin. 2012. Identifying high-impact sub-structures for convolution kernels in document-level sentiment classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers - Volume 2*, ACL '12, pages 338–343, Stroudsburg, PA.

Turney, Peter D. 2002. Thumbs up or thumbs down?: Semantic orientation applied to unsupervised classification of reviews. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, ACL '02, pages 417–424, Stroudsburg, PA.

Velikovich, Leonid, Sasha Blair-Goldensohn, Kerry Hannan, and Ryan McDonald. 2010. The viability of Web-derived polarity lexicons. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 777–785, Stroudsburg, PA.

Wainwright, Martin J. and Michael I. Jordan. 2008. Graphical models, exponential families, and variational inference. *Foundations and Trends in Machine Learning*, 1(1-2):1–305.

Wang, Sida and Christopher Manning. 2012. Baselines and bigrams: Simple, good sentiment and topic classification. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 90–94, Jeju Island.

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

Williams, Gbolahan K. and Sarabjot Singh Anand. 2009. Predicting the polarity strength of adjectives using Wordnet. In *ICWSM*, pages 346–349, San Jose, CA.

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

Yessenalina, Ainur, Yisong Yue, and Claire Cardie. 2010. Multi-level structured models for document-level sentiment classification. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1,046–1,056, Cambridge, MA.

Younger, Daniel H. 1967. Recognition and parsing of context-free languages in time n3. *Information and Control*, 10(2):189–208.

Yu, Hong and Vasileios Hatzivassiloglou. 2003. Towards answering opinion questions: Separating facts from opinions and identifying the polarity of opinion sentences. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, EMNLP '03, pages 129–136, Stroudsburg, PA.

Zelle, John M. and Raymond J. Mooney. 1996. Learning to parse database queries using inductive logic programming. In *AAAI/IAAI*, pages 1,050–1,055, Portland, OR.

Zettlemoyer, Luke S. and Michael Collins. 2007. Online learning of relaxed CCG grammars for parsing to logical form. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL-2007)*, pages 678–687, Prague.

Zettlemoyer, Luke S. and Michael Collins. 2009. Learning context-dependent mappings from sentences to logical form. 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 2 - Volume 2*, ACL '09, pages 976–984, Stroudsburg, PA.

Zhao, Jichang, Li Dong, Junjie Wu, and Ke Xu. 2012. Moodlens: An emoticon-based sentiment analysis system for Chinese tweets. In *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '12, pages 1,528–1,531, New York, NY.