

博士論文

**Sentiment Analysis
Using Polarity Bias and Correlation**

(感情極性の偏りと相関を用いた感情分析)

高文梁 (Gao Wenliang)

Graduate School of Information Science and Technology

University of Tokyo



Supervisor: Prof. Masaru Kitsuregawa

13th June 2014

Abstract

Sentiment Analysis Using Polarity Bias and Correlation (感情極性の偏りと相関を用いた感情分析)

高文梁 (Wenliang Gao)

Humans have the instinct to express their own feelings and observe others'. We learn to borrow others' sensation into our system and use them when similar situation occurs. Sentiment is about the feeling, the sensation and the opinion that have little to relate with absolute fact. The explosively growing of personal media, such as Twitter and Amazon, urges fast processing of sentiment that can never be possibly done with human resource. Sentiment analysis, as one of the most active research topic in NLP in recent years, studies statistical methods to process huge amount of sentiment hidden in text and makes it easier for us to borrow.

Sentiment classification is one of the fundamental tasks in sentiment analysis. It classifies a polarity label, e.g. positive or negative, for a given document. Traditional methods assume that the textual features are sufficient for this task. However, sentiment, held by a user and normally toward some target, varies drastically according to the user and the target. Therefore, some researches use user and target as supportive evidence for estimating the polarity. Unfortunately, these methods are often infeasible because the properties of users and targets are frequently unknown.

In this thesis, we propose two ways of modelling the user and target for sentiment classification. The key observations for these two methods are 1). the distribution of polarities given by each user or given to each target is often biased, and 2). users are correlated to each other and the targets are also correlated to each other. We encode these observations into global features that help the classifier understand the user and the target. By introducing the new global features, polarity labels became mutually depend on each other. To resolve dependencies among labels, we explore two approximated decoding algorithms, “easiest-first decoding” and “two-stage decoding.” Experimental results on real-world datasets confirmed that our methods contributed significantly to the classification accuracy.

Acknowledgements

I would never achieve a degree if without the help of many. I sincerely thank Prof. Masaru Kisturegawa, a great supervisor. He provided the best environment I can ever dream of, and guided me into the area of my current research. His extraordinary charisma and devoted attitude inspires me. He has plenty other aspects other than researches that I can learn. I am so grateful to Assoc. Prof. Masashi Toyoda. He kindly provided advises to countless things. Whenever I have a question, he is always available to discuss and gives constructive suggestions.

I give my special thanks to Assoc. Prof. Kaji and Assoc. Prof. Yoshinaga. They spent many hours to discuss with my researches. Their tremendous experience on sentiment analysis and natural language processing. I admire their professional skills in research and borrowed many from them. I also thanks the doctoral committees, Prof. Jun Adachi, Prof. Shuichi Sakai, and Prof. Kiyoharu Aizawa.

Many members in Kistsuregawa, Toyoda lab kindly lend me advices. Dr. Daisaku Yokoyama, Dr. Masahiko Itoh, Dr. Haichuan Shang and Dr. Rage Uday Kiran gave me plenty of advises to my presentations. Dr. Zhenglu Yang, Dr. Yongkun Wang and Dr. Yanhui Gu have helped me to blend into the new environment.

I am grateful to the Japanese government who funded me with MEXT

scholarship. The fund lends me 4 years to explore this study.

At last, I thank Ning Zhu for showing no borings when many times I introduce my work to you and Wula Gao for being with me all the way.

June 2014

Contents

Abstract	i
Acknowledgements	iii
List of Figures	ix
List of Tables	xiii
1 Introduction	1
1.1 Background	1
1.2 Problem	3
1.3 Method	4
1.4 Outline	6
2 Background	8
2.1 Sentiment Classification	8
2.1.1 Text-based Methods	9
2.1.2 User- or Product-Aware Methods	12
2.2 Graphical Model	13
2.2.1 Decoding	15
2.2.2 Graphical Model based Sentiment Analysis	19
2.3 Supervised Sentiment Classification	19

3	Modeling Polarity Bias	21
3.1	Motivation	21
3.2	Global Feature	22
4	Modeling Polarity Correlation	25
4.1	Motivation	25
4.2	Global Feature	26
4.3	Cluster of User and Target	28
5	Decoding Strategy	30
5.1	Global Dependency	31
5.2	Two Approximate Decoding Strategies	33
5.2.1	Easiest-first Decoding	33
5.2.2	Two-stage Decoding	35
5.3	Time Complexity	36
5.4	Training	37
6	Experiment	38
6.1	Setting	38
6.2	Datasets	39
6.2.1	A New Dataset: IMDB	39
6.2.2	Details	40
6.3	Result	42
6.3.1	Overall Accuracy	42
6.3.2	Accuracy of Bias Features	44
6.3.3	Accuracy of Clusters for Correlation Features	45
6.3.4	Accuracy of Decoding Strategy	49

7	Analysis	51
7.1	Speed and Accuracy of Decoding Strategy in Terms of Data Size	52
7.2	Analysis for Polarity Bias	55
7.2.1	Accuracy in terms of neighbor size	55
7.2.2	Polarity bias in terms of user or product	56
7.2.3	Impact of training reviews written by test users (or on test products)	59
7.2.4	Learning curves	61
7.2.5	Examples	62
7.3	Correlation Examples	64
7.3.1	None Clusterring	66
7.3.2	Clustered Users and Targets	70
8	Conclusion and Future Work	76
8.1	Conclusion	76
8.2	Future Direction	78
	Bibliography	80

List of Figures

2.1	Syntax parse tree of sentence “Finally joined team #iPhone5S.”. Symbol “S” at the top of the tree is the root of a sentence, “VP”, “ADV”, “RB”, “NP”, “VB”, and “NN” are terminal symbols denote verb phrase, adverb phrase, adverb, noun phrase, verb, and noun, respectively.	11
2.2	Independent model of tagging a given sentence with Part-of-Speech tags. Black lines represent dependencies.	14
2.3	Global model of tagging a given sentence with Part-of-Speech tags. Black lines represent dependencies.	15
2.4	Viterbi decoding on Example 2.2.1. Thick line is the chosen path by the max function in Equation 2.2.	16
2.5	Easiest-first decoding on the Example 2.2.1. In each iteration, it labels one word with the highest probability (bold font p). Note that the third step in the right bottom corner, the strategy uses the global classifier $p(y_n y_{n-1}, \mathbf{x}_n)$	17
2.6	Two-stage decoding on the Example 2.2.1. In the first stage, it labels each word with the local classifier ($p(y_n \mathbf{x}_n)$). In the second stage, it uses the label assignment in the previous stage as the condition and update the labels with global classifier ($p(y_n y_{n-1}, \mathbf{x}_n)$).	18

3.1	The user and target are frequently biased on giving sentiment polarities.	22
4.1	The history comments of the user and the target helps the current label estimation.	26
5.1	Global dependency among labels and global features. Black line means dependency.	31
5.2	Easiest-first strategy	34
5.3	Two-stage strategy	35
7.1	The classification accuracy computed accumulatively when we changed the size of testing reviews.	52
7.2	Average computation time when we changed the size of testing reviews.	52
7.3	The user bias distribution. The users who only have one review are eliminated.	57
7.4	The product bias distribution. The products who only have one review are eliminated.	58
7.5	Average accuracy when we change the size of training data. . .	62

List of Tables

6.1	Dataset statistics.	40
6.2	Accuracy (%) on review datasets. <i>bias</i> and <i>correlation</i> mean using the corresponding global features defined in chapter 3 and 4. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).	43
6.3	Accuracy (%) on review datasets. <i>+user</i> and <i>+target</i> mean using the corresponding global features defined as polarity bias in chapter 3. Accuracy marked with “ $>$ ” and “ \gg ” was significantly better than baseline ($p < 0.05$ and $p < 0.01$ assessed by McNemar’s test).	45
6.4	Accuracy (%) on dataset Pang when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.	46
6.5	Accuracy (%) on dataset Blitzer when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.	47
6.6	Accuracy (%) on dataset IMDB when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.	48

6.7	Accuracy (%) on three datasets when choosing different decoding strategies, namely easiest-first (e-f) and two-stage (t-s). <i>+bias</i> and <i>+correlation</i> mean using the corresponding global features defined as polarity bias in Chapter 3 and polarity correlation in Chapter 4.	49
7.1	Accuracy (% , lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the Pang dataset divided according to the number of reviews written by the user and the number of reviews on the product. The float inside parentheses is the difference from the baseline method. No accuracy is significantly different from baseline ($p \geq 0.01$ assessed by McNemar’s test).	53
7.2	Accuracy (% , lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the Blitzer dataset divided according to the number of reviews written by the user and the number of reviews on the product. The float inside parentheses is the difference from the baseline method. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).	54
7.3	Accuracy (% , lower inside cell) of proposed method (two-stage) and the review size (upper inside cell) on the Maas dataset divided according to the number of reviews on the product. The float inside parentheses is the difference from the baseline method. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).	55

7.4	Accuracy (%) on seen/unseen user or product splits of the Pang dataset. <i>su</i> , <i>uu</i> , <i>sp</i> and <i>up</i> stand for <i>seen user</i> , <i>unseen user</i> , <i>seen product</i> and <i>unseen product</i> respectively. No accuracy is significantly different from baseline ($p \geq 0.01$ assessed by McNemar’s test).	59
7.5	Accuracy (%) on seen/unseen user or product splits of the Blitzer dataset. <i>su</i> , <i>uu</i> , <i>sp</i> and <i>up</i> stand for <i>seen user</i> , <i>unseen user</i> , <i>seen product</i> and <i>unseen product</i> respectively. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).	60
7.6	Accuracy (%) on seen/unseen product splits of the Maas dataset. <i>sp</i> and <i>up</i> stand for <i>seen product</i> and <i>unseen product</i> . Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).	61
7.7	Examples show the influence of <u>leniency</u> and <u>popularity</u> global features. The bold content is the negative evidence learned by classifier. bl is baseline method.	63
7.8	The top ten correlation user features ($u_i^{y_i} - u_j$) in Pang dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).	64
7.9	The top ten correlation target features ($t_i^{y_i} - t_j$) in Pang dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).	65
7.10	The top ten correlation user features ($u_i^{y_i} - u_j$) in Blitzer dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).	66
7.11	The top ten correlation target features ($t_i^{y_i} - t_j$) in Blitzer dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).	67
7.12	The top ten correlation user features ($u_i^{y_i} - u_j$) in IMDB dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).	68

7.13	The top ten correlation target features ($t_i^{y_i} - t_j$) in IMDB dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).	69
7.14	The top ten correlation target (clustered) features ($tc_i^{y_i} - tc_j$) in Pang dataset with highest absolute weight ($w_{tc_i^{y_i} - tc_j}$). For each cluster, two most representative movies (separated by comma) and the approximate genre is given.	72
7.15	The top ten correlation user (clustered) features ($uc_i^{y_i} - uc_j$) in Blitzer dataset with highest absolute weight ($w_{uc_i^{y_i} - uc_j}$). For each user cluster, two most representative users are listed (separated by comma).	73
7.16	The top ten correlation target (clustered) features ($tc_i^{y_i} - tc_j$) in Blitzer dataset with highest absolute weight ($w_{tc_i^{y_i} - tc_j}$). For each cluster, two most representative products (separated by comma) and the approximate genre is given.	74
7.17	The top ten correlation target (clustered) features ($tc_i^{y_i} - tc_j$) in Imdb dataset with highest absolute weight ($w_{tc_i^{y_i} - tc_j}$). For each cluster, two most representative movies (separated by comma) and the approximate genre is given.	75

Chapter 1

Introduction

1.1 Background

Twitter nowadays has 400 million tweets written by its users in each day while that number was only 5,000 in 2007. The huge amount of content is generated by drastically growing users. According to statista.com, the number of active users grows from 30 millions in 2010 to 255 millions in 2014. While Twitter is only one example of plenty others, such as Amazon.com, Facebook or Google+. Different from traditional internet document that are created by officials, these documents are written by individual users to freely express their own experiences. The problem of how to understand the contents and how to process the huge amount is still haunting researchers.

Among all the proposals, sentiment analysis is one of the most active ones. On the contrary of factuality, sentiment is about each individual: whether you like or hate, happy or angry, boring or interested. Sentiment analysis is a perfect tool for processing such contents of personal experiences and providing the result to support things like guide users to make clever choices and lead companies to decide their strategies .

Human has the instinct of borrowing others' experience [Car05]. The more examples we learn and the higher quality these examples are, more competitive we became for thriving. Sentiment analysis automatically and statistically summarizes large-scale examples. It covers sentiment classification, subjectivity discrimination, target extraction and many other aspects. [PL08, Liu12]. Sentiment classification, which automatically assign a polarity label to a document, is considered a fundamental and important task. How to solve the problem with higher accuracy attracts researchers. Traditional sentiment classification categorize a given document by only the textual features [PLV02, Tur02]. For instance, given the following examples:

Example 1.1: I love my new **iPhone5s** (by user John),

Example 1.2: Finally joined team **iPhone5s** (by user Mary),

Traditional methods could easily classify Example 1.1 because word “*love*” has strong indication of positive polarity. But for Example 1.2, it is not so obvious because no strong indication toward any polarity in this example.

Human can easily estimate the polarity of Example 1.2 is positive because we have background knowledge that a female user (Marry) is normally eager to buy a **iPhone5S**. The definition of sentiment [Liu12] is a quintuple, (g, s, h, t) , where g is the sentiment target, s is the sentiment polarity, and h is the sentiment holder (user). This means that the sentiment is closely related to the user and the target, therefore we should introduce methods that exploit the usage of user and target.

1.2 Problem

Before going into further discussion of user and target aware methods, we should first study the real-world data we are treating with. A real review example is given in the following.

Review:

Product: iPhone 5s:

Made in USA or Imported

4.0-inch Retina display

A7 chip with M7 motion coprocessor

Touch ID fingerprint sensor...

By Daniel

Here is my advice, if you can buy a iPhone 5s from the Istore or try to buy it unlocked from any carrier do it because here in Amazon they promise you a unlocked phone for 100 more dollars than its original price and guess what, you get a locked phone that has a AT&T gsm card, i live here in Venezuela, i bought a new nano sim for my phone and it didn't work, if you are out of United States and you want a iPhone 5s i recommend you to wait for it to arrive to your local carriers.

The comments are given by a user and on a target (product in this case). Target is briefly described for the review. User name is mostly directly given. Sentiment target is explicitly given in the reviews, and normally implicitly

given in other materials (e.g. a hashtag of “iPhone5s”).¹ Sentiment classification labels a given document with positive or negative polarity. The language given by users are normally informal, for example in our review example there is only one period and all “*P*”s are not capitalised. Existing methods based on formal text can hardly classify such examples.

1.3 Method

Going back to Example 1.2, if the classifier knows “*Joined team iPhone5s*” means honour to Marry (user aware) or it knows that iPhone5s is a great product (target aware), then it can easily give a positive score.² Social network is a popular solution by referring to the ratings of given user’s friends [TLT⁺11, SSUB11, JYZ⁺11]. For example, classifier knows Marry’s friends all likes to join team iPhone5s. However, such social network only exists in a limited number of resources. An user- and target-specific classifier may assign different scores to textual features according to different users and targets. For example, Marry often use “finally” for positive sentiment or “join team” for iPhone5s to accompany with positive sentiment. However, to train such a classifier we need to label many training data for each user and target. Recommender system is naturally connected to sentiment analysis because they deal with ratings. However, resources like Twitter lack the ratings history but rich in comments.

In this thesis, two properties of user and target are proposed, namely

¹Sometimes, the target is not directly given, but we can extract it by existing methods [JG10, JYZ⁺11, QLBC11].

²In this thesis, I interchangeably use review and sentiment containing text, since the main type of document sentiment analysis treating with is review. Therefore, sentiment target and product means the same and user is the opinion holder.

polarity bias and polarity correlation. Both of the proposed properties help improve sentiment classification but neither is directly given. We study two decoding strategies which collectively compute the proposed properties. Our contributions are:

- We propose polarity bias and polarity correlation of users and targets to support sentiment classification. Each of the properties is modeled as novel features in a supervised classification framework. Although they improve the classification accuracy, they are not directly given.
- We build a global model to compute the proposed features. The newly introduced two kinds of global features introduce global dependencies between the labels. We firstly translate the new problem into classical inference problem on general CRFs model. Then two efficient decoding strategies, namely easiest-first [TT05] and two-stage [KM06], are deployed on our large-scale dependency model.
- Experimental results on four real-world data explain our advantage on accuracy of sentiment classification. Two of the datasets are existing large-scale corpus. We collected one review dataset from IMDB, which is the largest among the four datasets. Experiment shows that our proposals improve accuracy on the three larger datasets compared to existing baselines.
- A thorough analysis is performed at the end of this thesis.

Here, we mention global features that is different from local features with respect to whether the feature is accessible for the given document. For example, text is accessible for Example 1.1 and 1.2 but the user’s rating history is not accessible. Though in this thesis our methods are tested only

for sentiment classification, we consider it a pioneer for other researches in sentiment analysis which are also highly related to user and target.

1.4 Outline

We interchangeably use target and product for the same meaning because normally the target of a review is a product. The rest of this thesis is organised as follows.

Chapter 2 introduces the related studies to our work. The sentiment classification task is separately discussed by whether it concerns user and target. Then, we show the background problem of decoding method and how it is performed. At last, we discuss the supervised classifier framework that our methods adopt.

Chapter 3 and 4 show our two ideas, polarity bias and polarity correlations, of modelling user and target. The two ideas are both represented as global features in a supervised classifier. Polarity bias method is based on our observation that the ratings given by a user or given to a target are frequently biased toward one polarity. We call such properties user leniency and product popularity and describe each of them as two real valued features in a supervised classifier. Polarity correlation is based on our observation that targets and users have mutual connections, e.g. users who are friends tend to give similar ratings while users who are rivals tend to give opposed ratings. We describe this property by constructing a group of indicator features in a supervised classifier.

Two decoding strategies, easiest-first and two-stage, are discussed in Chapter 5. To solve the global dependencies introduced by new global features, we use approximate decoding strategies, which are comparatively fast.

In Chapter 6, we perform experiment to compare our accuracy with several existing methods. The results show that our proposals significantly outperformed the baselines on three large-scale datasets.

We give a thorough analysis in Chapter 7 and finally conclude in Chapter 8.

Chapter 2

Background

This section reviews the background knowledge. Sentiment analysis is brought up more than ten years ago, and still one of the most thriving topics. In this chapter, the fundamental task of sentiment analysis, sentiment classification, is introduced in the first. Then, we briefly survey the global dependency model and the decoding strategy which serves a important role in our work. At last, we introduce the supervised classification framework that is used by our proposals.

2.1 Sentiment Classification

Sentiment, on the opposite side of objectivity, represent the inner movement of a individual. With the growing attention of individual user's existence, analysis of such information has become attractive. Such research supplies personalised service to end users and enables companies to grasp their customers.

Sentiment analysis is devoted to study human's sentiment activity from a given material (mainly text). Deciding which kind of sentiment a given

text represent (also called sentiment classification) is especially vital in this area. It focuses on labelling a given document with a sentiment polarity [PL08, Liu10]. The most obvious polarity categorisation is “like” (positive) or “hate” (negative). There are also other kinds, e.g. “anger” and “surprise”, but here we only consider the negative and positive polarities for simplifying the problem. Other similar categorisations (e.g. rating score from one star to five stars) can be easily transformed into our two polarities setting.

Normally, the content of the document, the user who wrote the document, and the target on which the document is written are considered crucial to this research [PL08]. In what follows, we briefly glance at the traditional approaches based on purely textual content, and then introduce user- or product-aware approaches.

2.1.1 Text-based Methods

Most existing studies consider that only textual features are effective for classifying the sentiment [PL08, Liu10, Liu12]. Based on different perspectives, text-based sentiment classification can be categorised differently. Here, we discuss these methods according to which kind of textual features does the method use.

Lexical Features

The most straightforward features are lexical features, such as n -grams of the document. Taking the review example introduced in Section 1.2, the features are (“I”, “bought”, “an”, ..., “assist”, “.”). Pang *et al.* (2002) firstly brought up the supervised sentiment classification approach which takes word n -grams as features for a SVM classifier.

Compared to traditional bag-of-word features, tf-idf measures the feature

with respect to the local weight and the global importance [MF09, PT10]. Martineau and Finin (2009) designed delta tf-idf weighting that is shown to provide better feature vector. Paltoglou and Thelwall (2010) empirically measured the effectiveness of these features on sentiment classification task.

In early studies, sentiment lexicon has been tremendously useful for this task [Tur02]. One entry in such a lexicon contains a word face (e.g. “like”) and the positive and negative scores [HL04, WHS⁺05, AB06, BES10]. When classifying a document, positive and negative ratios of such entry word are informative features [Tur02, WWH09, TBT⁺11]. Most work uses a mutual information-based propagated method [ES07, GvDB12] or a syntactical relation-based bootstrapping method [QLBC11, HYW⁺14, ZS14] to construct a comprehensive lexicon.

Topic model, such as LDA [BNJ03], translates words into concepts with less dimensionality. Some recent studies extend LDA model to handle sentiment analysis [TM08, LH09, LHZ10, BGR10, JO11]. Lin and He (2009) assumed each word in the document is assigned with a topic. Jo and Oh (2011) assumed each sentence with a topic. Boyd-Graber and Resnik (2010) took advantage of topic consistency cross different languages.

Polarity Shifter

A polarity shifter is normally a modifier that shift the polarity strength or direction of a sentiment word [PZ04, KI06, WBR⁺10]. For example, “don’t like” reverses the direction and “very like” strength the polarity of word “like”.

Numerous studies under this scope can be differentiated by their assumption of the functionality of polarity shifter [LLC⁺10, TM08]. Some work assumes a polarity negator (one kind of polarity shifter) simply flips the

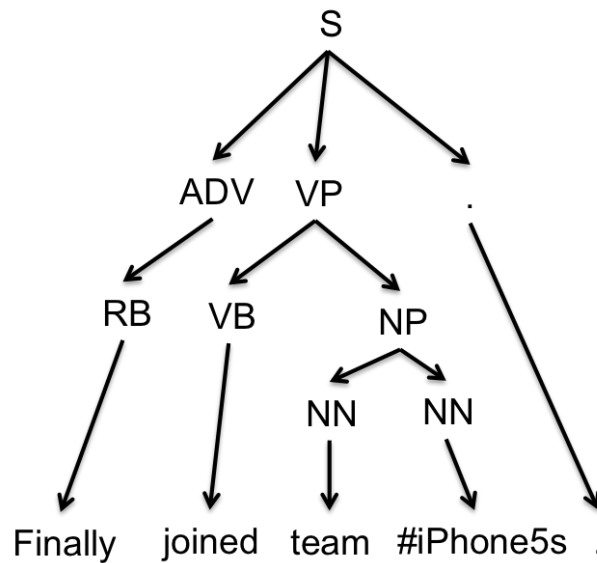


Figure 2.1: Syntax parse tree of sentence “Finally joined team #iPhone5S.”. Symbol “S” at the top of the tree is the root of a sentence, “VP”, “ADV”, “RB”, “NP”, “VB”, and “NN” are terminal symbols denote verb phrase, adverb phrase, adverb, noun phrase, verb, and noun, respectively.

polarity of the modified word [PZ04, KI06]. Another perspective assumes that shifters change a constant value of the polarity word [LS09, TBT⁺11]. Most work takes a supervised learning approach to determine the shifted polarity [JYM09, LRO12, BCM⁺12]. Zhu *et al.*, (2014) automatically detect the utility by modelling clausal conjunct word, such as “but” and “however” [ZGMK14].

Syntax Features

As shown in Fig. 2.1, syntax is the grammatical aspect of the language [Cho57]. Plenty of studies on sentiment analysis are related to the this topic. Here, they are separated into two categories based on their task, namely syntax based product feature extraction and polarity classification.

Product feature extraction focuses on finer-grained sentiment analysis [WZHW09, LHH⁺10, MW10, QLBC11, YC13, LXZ14]. Wu *et al.* (2009) implemented a SVM classifier taking syntactical features. Li *et al.* (2010) and Yang and Cardie (2013) used CRFs model. Qiu *et al.* (2011) designed several syntactical rules to extract sentiment target using bootstrapping method. Liu *et al.* (2014) took label propagation methods to rank target candidates.

Syntactic structure has great impact on the accuracy for sentiment polarity classification [WHS⁺05, KN06, ML07, MP07, MHN⁺07, JYM09, NIK10, TM11, ZLG⁺11, TE13, LTS13, YC14]. The simplest way is to take dependencies as features in supervised classifier [WHS⁺05, MP07, JYM09]. Some studies created latent sentiment variables in CRFs model based on the syntactic structure [MHN⁺07, NIK10]. Others argued that the sentential connectors, e.g. “but” and “and”, are informative indicator for the change of sentiment [KN06, ML07, TM11, ZLG⁺11, TE13, LTS13, YC14]

2.1.2 User- or Product-Aware Methods

Recently, user generated content attracts attention more than ever, which stimulates researchers to explore the effectiveness of user and product information. Tan *et al.*, (2011) and Speriosu *et al.*, (2011) exploited a user network behind a social media (Twitter in their case) and developed a graph-based method under the assumption that friends give similar ratings towards same products. However, such user networks are not always available in the real world datasets and when they exist the network may not relate to the target products domain.

Seroussi *et al.* (2010) computed the similarities among users on the basis of text and their rating histories. Then, they classified a given review by referring to the ratings given for the same product by other users who

are similar to the user in question. Li *et al.* (2011) incorporated user- or product-dependent n -gram features into a classifier. They argued that users use a user-specific language to express their sentiment, while products are also commented by a product-specific language. These approaches, however, assume that the training data contains reviews written by the test users or on the test products. This is an unrealistic assumption since we need to label reviews required for every emerging user or product.

In this study, we intend to handle reviews written by emerging users or on emerging products by capturing characteristics of such users or products from the test reviews. As we later confirm in experiments, our method improves the classification accuracy in almost all circumstances.

2.2 Graphical Model

Graphical modelling is a powerful tool for representation and decoding in multivariate probability distributions [SM12]. Among all the models, we are interested one common kind that given a set of feature vectors, $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_N\}$, to predict a label vector, $\mathbf{y} = \{y_1, y_2, \dots, y_N\}$. Each feature vector, \mathbf{x}_n , represents the information of instance n . y_n is the label we are targeting. For example, we want to estimate the Part-of-Speech tags of a given sentence [JM09] as:

Example 2.2.1 *Book that flight.* \rightarrow *Book/VB that/DT flight/NN ./.*

where *VB* denotes verb, *DT* denotes determiner and *NN* denotes noun.

Depending on how we assume the dependencies, it can be modelled differently. Given Example 2.2.1, we can assume the Part-of-Speech tags only depend on the corresponding word, $p(y_n|\mathbf{x}_n)$ [Mar83, TKMS03], where $p()$

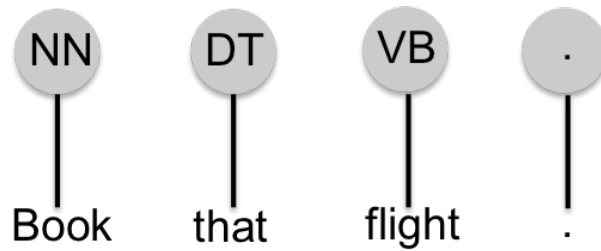


Figure 2.2: Independent model of tagging a given sentence with Part-of-Speech tags. Black lines represent dependencies.

is the probability function. The graphical model is shown as Fig. 2.2. Obviously, the tag for “Book” and “flight” is wrong here since the classifier independently computes the probability of the two words and found that “Book” has been more used as a noun and “flight” has been more used as a verb.

The problem here is when the classifier estimates the tag, it independently looks at the local word. If we provide global information of the neighbours’ label, classifier will gain more advantage on deciding the current word’s label [Kup92, TKMS03]. For instance as shown in Fig. 2.3, when labelling word “Book”, our probability became $p(y_n|y_{n-1} = START, \mathbf{x}_n = Book)$, where *START* denote the start of a sentence. A more generalised form is $p(y_n|y_{n-1}, \mathbf{x}_n)$, which means to estimate the POS tag of n th word, given the word n (e.g. “book”), the POS tag of the previous word (e.g. *START*). Thus, the classifier would successfully label “flight” as *NN* and label “Book” as because “flight” after a determinate (*DT*) is more likely to be a noun and “Book” at the start of a sentence is more likely to be a verb (*VB*).

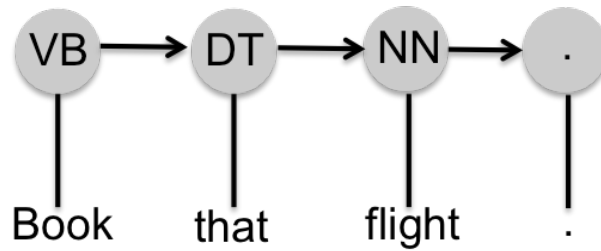


Figure 2.3: Global model of tagging a given sentence with Part-of-Speech tags. Black lines represent dependencies.

2.2.1 Decoding

After defining the global model, the question is how to choose a set of labels \mathbf{y} that maximises probability denoted as $\arg \max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x})$. The way of choosing a maximum configuration of \mathbf{y} with the consideration of dependencies between labels to is called decoding.

One simple way is to adopt a exact decoding strategy. In this strategy, machine is programmed to try every possible assignment to the labels. Assuming the possible assignment of a word $y_n \in \{NN, DT, VB, .\}$ in the example 2.2.1, the length of the sentence is N , then exact decoding's total time complexity is $O(M^N)$, where M is the size of possible assignment and N is the number of the instances. Sometimes the number of instances could be very large, for example, english sentence can be unlimitedly long by adding modifier clauses or conjunctions.

Viterbi:

For estimating a label configuration \mathbf{y} , we use Bayesian rule to decompose the probability according to the graphical model in Fig. 2.3. Viterbi is one of the most deployed algorithms for decoding on chain model. To perform the decoding we first rewrite the probability function by introducing viterbi

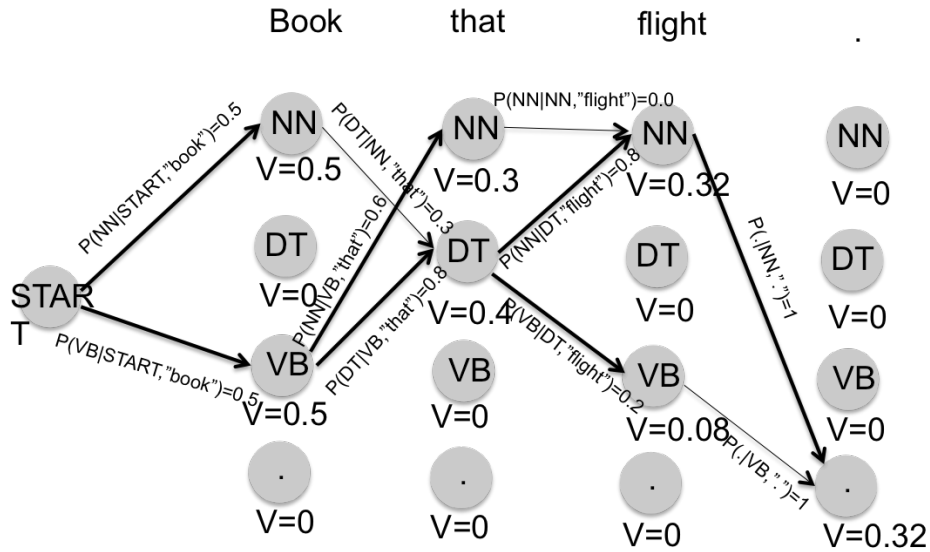


Figure 2.4: Viterbi decoding on Example 2.2.1. Thick line is the chosen path by the max function in Equation 2.2.

function $v()$ [SM06, JM09]:

$$p(\mathbf{y}|\mathcal{X}) = \max_{y_N} v(y_N), \quad (2.1)$$

$$\text{where, } v(y_n) = \max_{y_{n-1}} p(y_n|y_{n-1}, \mathbf{x}_n)v(y_{n-1}), \quad (2.2)$$

$$v(y_1) = p(y_1|START, \mathbf{x}_1). \quad (2.3)$$

Pick the assignment of the chain model, it pick the maximum $v(y_N)$ and pick the previous assignment along with the chain using the function 2.2. $p(y_n|y_{n-1}, \mathbf{x}_n)$ is the transmission probability from y_{n-1} to y_n given the observed word \mathbf{x}_n . As illustrated in Fig. 2.4, viterbi decoding firstly computes the $v()$ scores for each label of each possible assignment. It then traverses backward from the end of the sentence.

The problem of Viterbi decoding is that the model must contains no circle. Sometimes we need looped model to perform more accurately, for instance

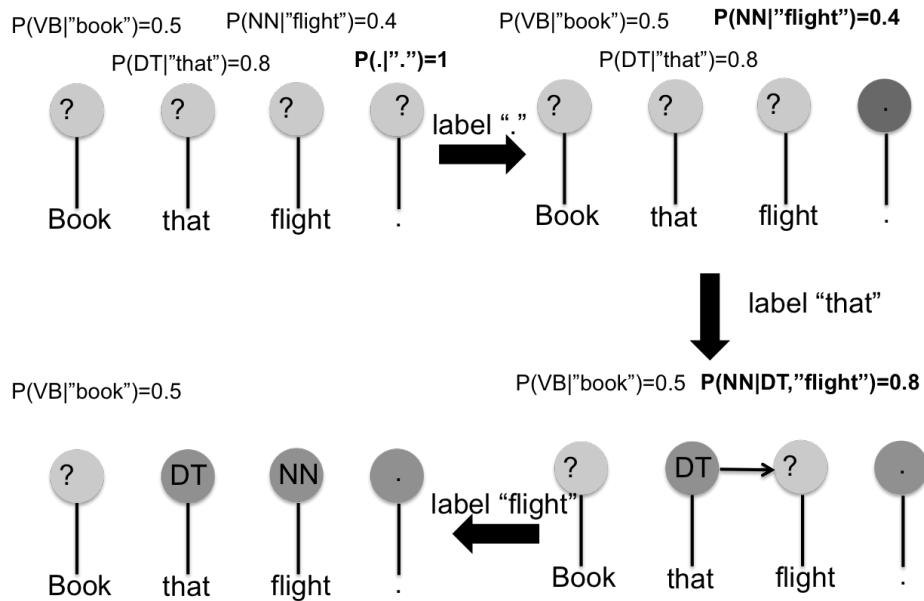


Figure 2.5: Easiest-first decoding on the Example 2.2.1. In each iteration, it labels one word with the highest probability (bold font p). Note that the third step in the right bottom corner, the strategy uses the global classifier $p(y_n|y_{n-1}, \mathbf{x}_n)$.

adding backward dependencies in model shown in Fig. 2.3. Viterbi cannot decode on such model since it pick the maximum of y_N which requires the table in Fig. 2.2 to be constructed.

Easiest-first:

Tsuruoka and Tsujii (2005) proposed to pick the easiest label to assign at each time. By “easiest”, they mean the assignment is most confident. The problem of this proposal is that sometimes, the decoding strategy need to estimate the label without global knowledge (e.g. y_{n-1} is unknown). To solve the problem, they constructed both a local classifier ($p(y_n|\mathcal{X})$) and a global classifier ($p(y_n|y_{n-1}, \mathcal{X})$).

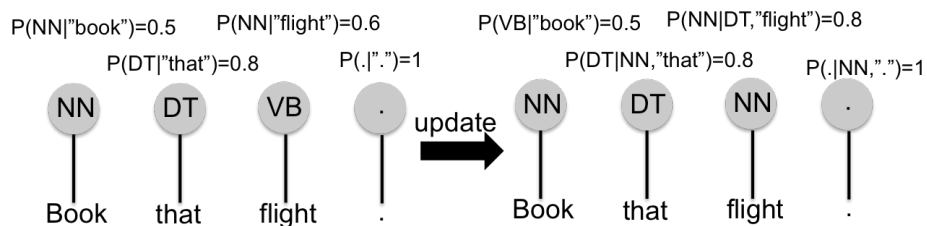


Figure 2.6: Two-stage decoding on the Example 2.2.1. In the first stage, it labels each word with the local classifier ($p(y_n|\mathbf{x}_n)$). In the second stage, it uses the label assignment in the previous stage as the condition and update the labels with global classifier ($p(y_n|y_{n-1}, \mathbf{x}_n)$).

This decoding method can handle loopy dependencies by ignore the global variables if they are not estimated yet. When the label is estimated, it never changes the label which makes easiest-first a relatively fast algorithm. However, to compute the maximum confident value, it maintains a heap structure that could hinder the speed. We give further introduction of this decoding method in Chapter 5 and analyse it in Chapter 7.

Two-stage:

Krishnan and Manning (2006) introduced to use the tentative label assignment for global variables as shown in Fig. 2.6. They divide the process into two steps. In the first step, they compute a approximate estimation with the local classifier. In the second step, they update the estimation by using the global classifier where the global variables are assigned by the first step's estimation.

This decoding method seems to estimate a more coarse work, but performs very fast. A detailed introduction and a analysis are given in Chapter 5 and Chapter 7, respectively.

2.2.2 Graphical Model based Sentiment Analysis

Existing graphical model based approaches used graphical model to describe dependencies inside a document [MHN⁺07, NIK10, DS11, SPH⁺11, SPW⁺13, YC14]. Duric and Song (2010) adopted HMM model to select features for sentiment classification. McDonald *et al.* (2007), Nakagawa *et al.* (2010) and Yang and Cardie (2014) deployed a CRFs based classifier. Socher *et al.* (2011), Socher *et al.* (2013) and Zhu *et al.* (2014) built a hierarchical model.

The greatest difference between our proposals and the existing ones is scale of dependencies. We assume the dependencies between documents while they assume that between words inside a sentence, or sentences inside a document. The number of the documents is normally over thousands but the number of words in a sentence is normally under 100. How to decode on a model with huge number of dependencies is a challenge in our study.

2.3 Supervised Sentiment Classification

Sentiment classification normally deal with predicting a label of a given document. More specifically, given a set of N documents \mathcal{D} , our task is to estimate label assignments $\hat{\mathcal{Y}}$, where $\hat{y}_n \in \{+1, -1\}$ for each given document $d_n \in \mathcal{D}$, $+1$ and -1 represent positive and negative polarity, respectively. The document d_n is a triple (doc_n, u_i, t_j) , where doc_n is the text, u_i is the user who wrote the document, and t_j is the target. The label of each review is estimated based on the following scoring function,

$$score(\mathbf{x}_n) = \mathbf{w}^T \mathbf{x}_n, \quad (2.4)$$

where \mathbf{x}_n is feature vector representation of the document d_n and \mathbf{w} is a weight vector which we learn from labeled data. Feature vector $\mathbf{x}_n =$

$\phi(doc_n, u_i, t_j)$, where $\phi()$ is the feature function that previously defined.

The weight parameters are learned by optimising a predefined loss function (e.g. $\ell(\mathbf{w}) = \sum_{n=1}^N \ln(y_n - \mathbf{w}^T \mathbf{x}_n)^2$), where y_n is the gold label of document i [Bis06]. With this scoring function, the label is estimated as follows:

$$\hat{y}_r = \text{sgn}(\text{score}(\mathbf{x}_n)) = \begin{cases} +1 & \text{if } \text{score}(\mathbf{x}_n) > 0, \\ -1 & \text{otherwise.} \end{cases}$$

Now the question is how to define feature function $\phi()$ with respect to user u_i and target t_j . We first split the feature function into two parts, local and global features as:

$$\phi(doc_n, u_i, t_j) = (\phi_{local}(doc_n), \phi_{global}(u_i, t_j)) \quad (2.5)$$

The first part in the RHS of Equation 2.5 extracts features from local text, e.g. ngram features. The second part in the RHS of Equation 2.5, called global features, considers the features to represent user u_i and target t_j . The textual features can take any form which only involves the text, e.g. dependency relation of the words. The left problem is how we define the global part of the feature function.

The following sections give two definitions based on different observations. The first one takes advantage of inner properties of user and target. The second one uses intra-relations (correlation) between users and between targets.

Chapter 3

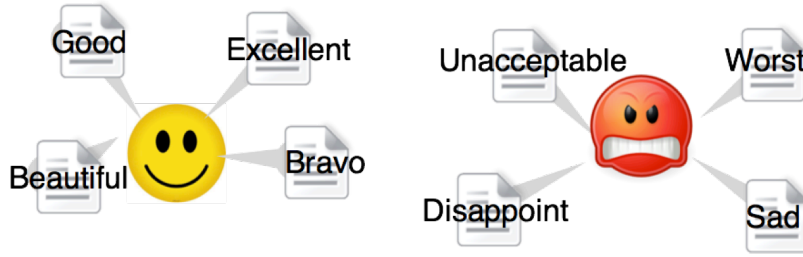
Modeling Polarity Bias

In this section, we discuss the polarity bias properties of users and targets. This representation are designed in favour of supporting sentiment classification. We first give the motivation and then we define it as global features in Section 3.2 under the supervised classification framework given in Section 2.3

3.1 Motivation

Human judges with different standards. People would not be even to give same number of positive ratings and negative ratings. On the contrary, people are normally extremely unbalanced, for example, some persons are nice (more positive ratings) and the others are picky (more negative ratings). Targets, on the other hand, also reveal such properties that some targets easily receives positive comments and others are unlucky. For example, “Macbook” will be normally favoured over a worn desktop computer. Figure 3.1 shows the realistic examples of biased users and biased targets.

The properties are denoted as user leniency and product popularity here. Compared to another representation introduced in Section 4, they are inner



(a) Examples of biased users.



(b) Examples of biased targets

Figure 3.1: The user and target are frequently biased on giving sentiment polarities.

property that only concerns a single user or target. Classifier shift the scores according to the properties.

3.2 Global Feature

As equation 2.5, our features can be divided into local and global ones such that $\mathbf{x}_n = (\phi_{local}(doc_n), \phi_{global}(u_i, t_j))$. The local features ($\phi_{local}(doc_n)$) are conventional word n -grams ($n = 1$ and $n = 2$) with binary values that indicate the existence. The global features ($\phi_{global}(u_i, t_j)$) are the user leniency and product popularity that are represented as real values.

Given a document denoted as a triple (doc_n, u_i, t_j) , our global features are decomposed as:

$$\Phi_{global}(u_i, t_j) = (f_{-}u^+(u_i), f_{-}u^-(u_i), f_{-}t^+(t_j), f_{-}t^-(t_j)),$$

where $f_{-}u^+(\cdot)$ and $f_{-}u^-(\cdot)$ are the positive and negative ratio of labels that given by a user and $f_{-}t^+(\cdot)$ and $f_{-}t^-(\cdot)$ are the positive and negative ratio of labels that written on a given target. They are computed as

$$\text{User leniency} \begin{cases} f_{-}u^+(u_i) = \frac{|\{d_k | \hat{y}_k = +1, d_k \in \mathcal{S}_u(d_n)\}|}{|\mathcal{S}_u(d_n)|}, \\ f_{-}u^-(u_i) = \frac{|\{d_k | \hat{y}_k = -1, d_k \in \mathcal{S}_u(d_n)\}|}{|\mathcal{S}_u(d_n)|}, \end{cases} \quad (3.1)$$

$$\text{Product popularity} \begin{cases} f_{-}t^+(t_j) = \frac{|\{d_k | \hat{y}_k = +1, d_k \in \mathcal{S}_t(d_n)\}|}{|\mathcal{S}_t(d_n)|}, \\ f_{-}t^-(t_j) = \frac{|\{d_k | \hat{y}_k = -1, d_k \in \mathcal{S}_t(d_n)\}|}{|\mathcal{S}_t(d_n)|}. \end{cases} \quad (3.2)$$

Here, $\mathcal{S}_u(d_n)$ is the user-related neighbor set of $d_n = (doc_n, u_i, t_j)$, which contains the comments written by the same user u_i , while $\mathcal{S}_t(d_n)$ is the product-related neighbor set of d_n , which contains comments written on the same target t_j as d_n . If $\mathcal{S}_u(d_n)$ (or $\mathcal{S}_t(d_n)$) equals to 0, we set $f_{-}u^+(u_i)$ and $f_{-}u^-(u_i)$ (or $f_{-}t^+(t_j)$ and $f_{-}t^-(t_j)$) to be 0.

We use $f_{-}u^+(u_i)$ and $f_{-}u^-(u_i)$ to capture user leniency, i.e., how likely the user writes positive and negative reviews, respectively. While we also use $f_{-}t^+(t_j)$ and $f_{-}t^-(t_j)$ to capture product popularity, i.e., how likely positive and negative reviews are written on the product, respectively.¹ When $f_{-}u^+(u_i) = 1$ and $f_{-}u^-(u_i) = 0$, user u_i is a lenient person. Otherwise, the

¹ Considering $f_{-}u^+$ and $f_{-}u^-$ ($f_{-}t^+$ and $f_{-}t^-$) always sum up to 1, we could also use only one feature for each of leniency and popularity (e.g. $f_{-}u^+$ and $f_{-}t^+$). We ran some experiments and found out that the two features designation outperformed the one feature designation ($f_{-}u^+$, $f_{-}t^+$) on two out of three datasets (Maas and Blitzer). Such that we here choose to represent user leniency and product popularity by using both positive and negative ratio of labels.

user is a critical person. When $f_{-t^+}(t_j) = 1$ and $f_{-t^-}(t_j)$, target t_j is a popular target. Otherwise, it is not.

Chapter 4

Modeling Polarity Correlation

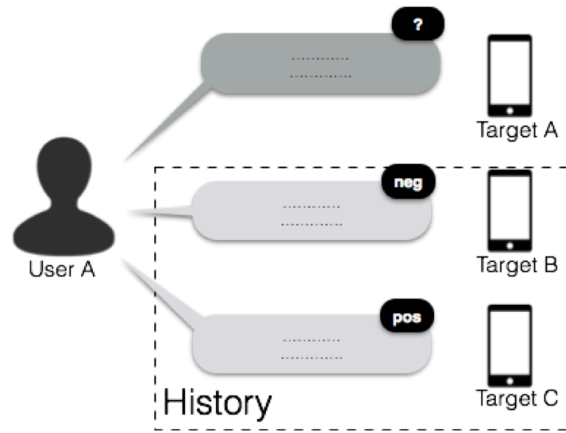
In this section, we discuss our second proposal for user and target properties, namely polarity correlation.

4.1 Motivation

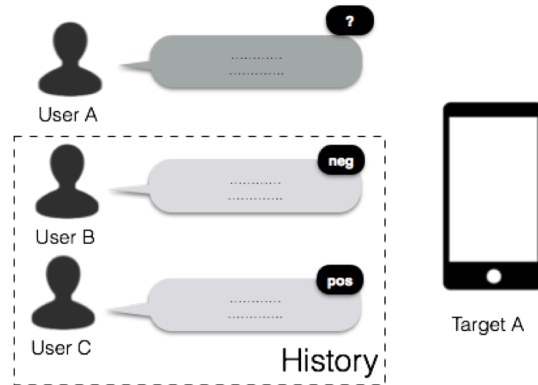
Sentiment targets are correlated such as “iPhone” and “Macbook” are always favoured by the same user but “iPhone” and “Surface” are rarely favoured by the same user. Similar phenomena also appears among users, e.g. “Steve Jobs” and “Bill Gates” have different tastes of product design.

With the consideration of the correlations, classifier gains useful evidence to further understand the user and the target of the given comment. The task of sentiment classification is to predict a polarity of the give comment, which is a triple (doc_n, u_i, t_j) .¹ As shown in Fig. 4.1, we describe the *user* and the *target* by the historical ratings. The problem now is how are we going to use the history and how the history is computed since the *user* or

¹ We can easily identify the user and the target of a given comment. When the target is not given, existing methods of target extraction [SME06] could be adopted.



(a) User history



(b) Target history

Figure 4.1: The history comments of the user and the target helps the current label estimation.

target may be unknown.

4.2 Global Feature

Our global features are responsible of providing user and target properties based on the polarity correlations. The correlations between two targets, t_i and t_j reflected by a users' ratings are denoted as:

- Friends:
 - likes $t_i \Leftrightarrow$ likes t_j ,
 - dislikes $t_i \Leftrightarrow$ dislikes t_j
- Enemies:
 - likes $t_i \Leftrightarrow$ dislikes t_j ,
 - dislikes $t_i \Leftrightarrow$ likes t_j
- Inclusive friends:
 - likes $t_i \Rightarrow$ likes t_j ,
 - dislikes $t_i \Rightarrow$ dislikes t_j
- Inclusive enemies:
 - likes $t_i \Rightarrow$ dislikes t_j ,
 - dislikes $t_i \Rightarrow$ likes t_j .

The correlations of two users reflected by a targets' ratings have the same categories. Our global features should be able to differentiate all kinds of correlations, thus using traditional similarity measurement [TLT⁺11, SSUB11] is insufficient.

Given an instance (doc_n, u_k, t_j) , we represent user u_k with all the rating history of u_k , $\{(t_i, y_l) | (doc_l, u_k, t_i) \in \mathbf{D}\}$. Each piece of the history, (t_i, y_l) , conditioned on the current target t_j is used as our global feature such as $t_i^{y_l} _ t_j \in \{1, 0\}$. Together with the corresponding weight, this feature explains the above correlations as:

- Friends:
 - $w_{t_i^+ _ t_j} > 0$ and $w_{t_j^+ _ t_i} > 0$,
 - $w_{t_i^- _ t_j} < 0$ and $w_{t_j^- _ t_i} < 0$
- Enemies:
 - $w_{t_i^+ _ t_j} < 0$ and $w_{t_j^+ _ t_i} < 0$,
 - $w_{t_i^- _ t_j} > 0$ and $w_{t_j^- _ t_i} > 0$
- Inclusive friends:
 - $w_{t_i^+ _ t_j} > 0$,
 - $w_{t_i^- _ t_j} < 0$

- Inclusive enemies: $w_{t_i^+ - t_j} < 0$,
 $w_{t_i^- - t_j} > 0$,

where the value of $w \in (-\infty, +\infty)$ is empirically learned using training data. We use the same way to construct user correlation global features, $u_i^{y_i} - u_j \in \{1, 0\}$.

To be noted here that the definition of $\mathcal{S}_u(u_i)$ and $\mathcal{S}_t(t_j)$ maintains to mean the neighbours of document $d_n = (doc_n, u_i, t_j)$. Now, the features vector \mathbf{x}_n is divided into local textual features and global correlation features: $\mathbf{x}_n = (\mathbf{x}_n^{text}, \mathbf{x}_n^{correlation})$.

4.3 Cluster of User and Target

There is one problem that features designed as a combination of two entities (targets or users) would be sometimes sparse. For example, target t_k is not contained in the training data, then no weight parameter would be learned for features $t_k^+ - t_?$, $t_k^- - t_?$, $t_?^+ - t_k$, and $t_?^- - t_k$, where $t_?$ denote any target. Rear features also suffer from insufficient learning examples problem.

Our simple solution is, cluster the targets and users. The clustered category assignment replaces the real user and target. More specifically speaking, the history of user and the history of target now is replaced with a set of features, $kt_i^{+/-} - kt_j$ and $ku_i^{+/-} - ku_j$. kt_i and kt_j are the assigned cluster of targets t_i and t_j , respectively. Notice that, the cluster assignment only replace the history that represent the user or target.

The new features have similar meanings as discussed in Section 4.2. After replaced with the cluster assignments, we are now trying to capture the correlation between clusters. The “friend”, “enemy”, “inclusive friend” and “inclusive enemy” relationships are about the two user clusters or two target

clusters.

The choice of number of clusters is a delicate parameter. If we choose it to be too small the positive history feature, $kt_i^+kt_j$, and the negative history features, kt_i-kt_j , would often appear together. This situation causes the distraction of classifier since like and dislike the same target seems absurd. Otherwise, if we choose the number to be too large, the sparsity problem would remain. Experiment shown in Section 6.3.3 measures the accuracy while changing different choice of cluster number.

Chapter 5

Decoding Strategy

In the previous chapters we discussed the two kinds of global features based on the two observations. They provide additional information to supervised classifier. Here, we assume a supervised classifier estimate polarity for a given document $d_i = (text_i, u_i, p_i)$, where $text_i$ is the content, u_i is the user, and p_i is the target. Our mission is to estimate a sentiment polarity assignment \mathbf{y} to a given set of test data \mathcal{D} , where $\mathbf{y}_i \in \{+, -\}$. We have to

1. extract the feature vector \mathbf{x}_i for each d_i ,
2. learn the weight parameter \mathbf{w} from the training data.

The feature vector is divided into local features and global features as $\mathbf{x}_i = (\mathbf{x}_i^{local}, \mathbf{x}_i^{global})$, where $\mathbf{x}_i^{local} = \phi(text_i)$ and $\mathbf{x}_i^{global} = \phi(u_i, p_i)$. The $\phi()$ is the feature extraction function. Here we the local feature vector is freely designed as long as it only involves text $text_i$. The global feature vector is constrained to picking either or both we discussed in Chapter 3 and 4. These global features are unknown if we naively implement the classification. In this section, we discuss the global model of our method.

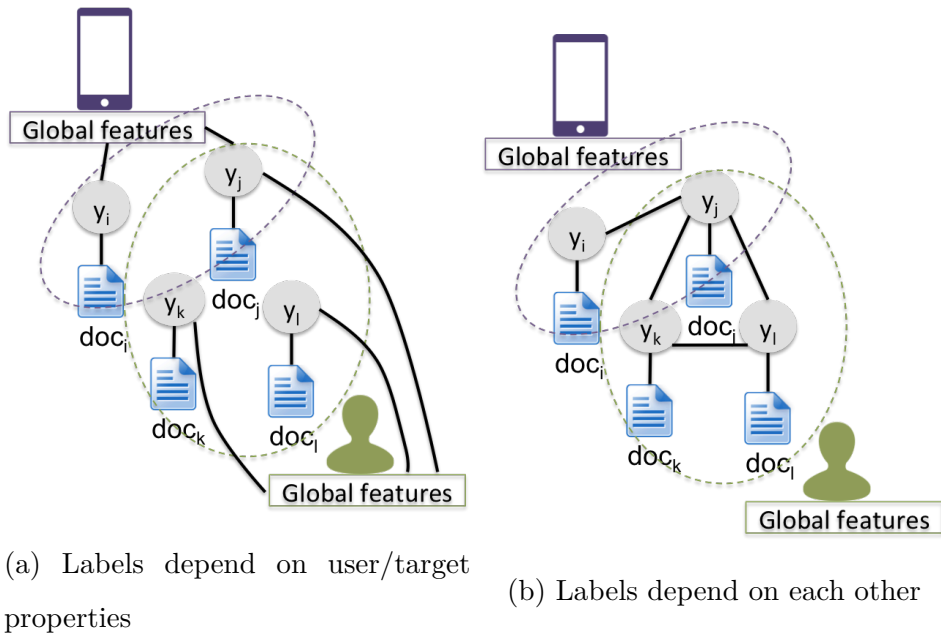


Figure 5.1: Global dependency among labels and global features. Black line means dependency.

5.1 Global Dependency

The global features we discussed in Chapter 4 declare dependencies between reviews. We compute each review’s label by using global features additional to local features. Thus the labels depend on global features. To get global features, our methods need to compute the labels first. Thus the global features also depend on the labels that belongs to other documents. Fig. 5.1a depicts the dependencies between global features and document labels. Because the global features are unknown, they introduce dependencies [KF09] such that we can simplify it as mutual dependencies between document labels, as shown in Fig. 5.1b.

The dependency in Fig. 5.1b is similar to the traditional problem on graphical model. The difference is the level of global dependency: previous

work deal with dependencies inside a document while ours deal with dependencies between documents. Loops of dependencies exist in our model, such that dynamic programming based decoding is infeasible. Since the number of dependency is in scale of the size of dataset, loopy belief propagation, which takes large amount of time to adjust labels, is also infeasible [MsJ99]. A scalable decoding strategy needs to be deployed to untangle the dependency. In the following, we consider three questions:

- Which decoding strategy we deploy,
- The time complexity of the decoding strategy
- How to train classifier in such decoding strategy.

Before further discussion, I denote two symbols here. Given a document $d_n = (text_n, u_i, t_j)$,

$$\mathcal{S}_u(d_n) = \{d_m = (text_m, u_k, t_f) | u_k = u_i, m \neq n, d_m \in \mathcal{D}\}$$

$$\mathcal{S}_t(d_n) = \{d_m = (text_m, u_k, t_f) | t_f = t_j, m \neq n, d_m \in \mathcal{D}\},$$

where $\mathcal{S}_u(d_n)$ denotes the document set which share the same user as d_n and $\mathcal{S}_t(d_n)$ denotes the document set which share the same target as d_n . In both proposed methods in Chapter 3 and 4, if the status of the user u_i updated, the score computed for each document in $\mathcal{S}_u(d_n)$ should be updated for the new global features. If the status of target t_j updated, all documents in $\mathcal{S}_t(d_n)$ should also be updated. Note that the updating of user and target status means a bit different for the two proposals. For the polarity bias proposal, this means the leniency or popularity value have changed. For the polarity correlation proposal, it means different history of the user or the target (adding new features or changing features).

5.2 Two Approximate Decoding Strategies

The global features make it difficult to perform decoding (i.e., labeling reviews) since each document can no longer be labeled independently. Exact decoding algorithms based on dynamic programming are not feasible in our case, because the search space grows exponentially as the number of test reviews increases. So instead, we explore and empirically compare two approximate algorithms: easiest-first [TT05] and two-stage [KM06] decoding strategy.

The major difference of the two algorithms is the way they use to compute global features. Easiest-first strategy uses the labels estimated by local features and previously computed global features, while two-stage strategy uses labels estimated by only the local features.

According to different global features (bias or correlation), the computation means a little differently. For polarity bias global features, when some review is newly labeled, the user’s leniency features should be recomputed using Equation 3.1 and the target’s popularity features should be recomputed using Equation 3.2. For polarity correlation global features, when some review is newly labeled, the user’s history and the target’s history (Figure 4.1) are increased thus new global features should be created.

We expect the easiest-first decoding will exhibit a better classification accuracy over the two-stage strategy, which we will confirm later in experiments. The easiest-first decoding is slower but expected to be more accurate, while the two-stage decoding is faster but expected to be less accurate.

5.2.1 Easiest-first Decoding

Fig. 5.2 depicts the easiest-first decoding algorithm. This strategy itera-

Input: A set of document \mathcal{D}

Output: Polarity estimation \mathbf{y}

- 1: **for** $d_i \in \mathcal{D}$ **do**
- 2: initialize the global features to 0
- 3: compute $\mathbf{y}_n = \text{score}(\mathbf{x}_n)$
- 4: **while** $\mathcal{D} \neq \emptyset$ **do**
- 5: $d_{max} = \arg \max_{d_i \in \mathcal{D}} |\text{score}(\mathbf{x}_i)|$
- 6: $y_{d_{max}} = \text{sgn}(\text{score}(\mathbf{x}_{d_{max}}))$
- 7: **for** $d_j \in (\mathcal{S}_u(d_{max}) \cup \mathcal{S}_t(d_{max})) \cap \mathcal{D}$ **do**
- 8: update global features
- 9: re-compute $\text{score}(\mathbf{x}_j)$
- 10: $\mathcal{D} = \mathcal{D} \setminus \{d_{max}\}$
- 11: **return** \mathbf{y}

Figure 5.2: Easiest-first strategy

tively determines the label of each document one by one. In each iteration, a document that is the easiest to label, i.e., the document d_i with the highest absolute score, $\text{score}(\mathbf{x}_n)$, is picked (line 5 in Fig. 5.2), and then we determine its label (line 6 in Fig. 5.2). This process is repeated until all the reviews are labeled. The global features are incrementally updated by using the labels of reviews that are already assigned. The updating of global features, to be more specifically, is recomputing the polarity bias global features and/or add newly discovered correlation feature. That is, at the beginning of decoding, all the global features are set to 0; as the labeling process proceeds, the global features become more accurate as more labels could be used to

Input: A set of document \mathcal{D}

Output: Polarity estimation \mathbf{y}

```
1: for  $d_n \in \mathcal{D}$  do
2:    $\mathbf{x}_n = \mathbf{x}_n^{local}$ 
3:    $\mathbf{y}_n = \text{sgn}(\text{score}(\mathbf{x}_n))$ 
4: for  $d_n \in \mathcal{D}$  do
5:   compute global features  $\mathbf{x}_n^{global}$ 
6:    $\mathbf{x}_n = (\mathbf{x}_n^{local}, \mathbf{x}_n^{global})$ 
7:    $\mathbf{y}_n = \text{sgn}(\text{score}(\mathbf{x}_n))$ 
8: return  $\mathbf{y}$ 
```

Figure 5.3: Two-stage strategy

compute them.

5.2.2 Two-stage Decoding

Fig. 5.3 depicts a two-stage decoding algorithm [KM06]. This strategy performs decoding twice. In the first stage (line 1 to line 3 in Fig. 5.3), we use only local features to classify the reviews. In the second stage (line 4 to line 7 in Fig. 5.3), those labels are used to compute global features and the labels are re-assigned by additionally using the computed global features. In our case, two-stage decoding at first only uses word n -gram features to estimate the labels. Thereafter, those labels are used to compute global features in the second stage.

5.3 Time Complexity

We here analyzes the time complexity of the two decoding strategies with respect to the number of test reviews, N . The time consumption of each polarity estimation is constant since there are much less global features than local features. Then the factors that concerned are how many such estimation is performed and the maintenance of data structure.

In easiest-first strategy, two processes consume most of the computation time. One is choosing the easiest review to label (line 5 in 5.2). The *arg max* operation spends $O(\log N)$ time in each iteration by using a heap structure to maintain the scores. Thus, the time complexity of this step is $O(N \log N)$ for N iteration. Another bottleneck is score re-computation (line 9 in 5.2). To update the score for each review $r \in \mathcal{S}_u(d_{max}) \cap \mathcal{S}_t(d_{max})$, we need $|\mathcal{S}_u(d_{max}) \cap \mathcal{S}_t(d_{max})|$ times *delete* and *insert* operations to the heap. If we could assume the number of reviews for each user or each product, $|\mathcal{S}_u(d_{max}) \cap \mathcal{S}_t(d_{max})|$ can be upper-bounded by a constant C .¹ The overall time complexity sums up to $O(N(\log N + C \log N)) = O(N \log N)$.

In the two-stage strategy, the complexity is $O(N)$ for both stages. Then the total complexity is also $O(N)$, which is the same as the existing method that uses only local textual features.

One more thing should be mentioned here. If the global dependencies exist across a larger range, e.g. all the document using word “good” are dependent on each other, the easiest-first consumes much more time because it needs to update more scores at each iteration.

¹However, based on our experiment as shown in Fig. 7.2, the number $|\mathcal{S}_u(d_{max}) \cap \mathcal{S}_t(d_{max})|$ is weakly related to N .

5.4 Training

It is straightforward to train the parameters of the scoring function for the two decoding algorithms if the classification process is assumed to be independent. We train a binary classifier as the score estimation function in Eq. 2.4, considering local features and global features. The values of global features are computed by using the gold labels. Which means, at the start of the training, different from testing, every label or the document is known. As the correlation global features might grow when new targets and new user appear, our classifier would ignore those global features unseen in the training. This classifier is used for easiest-first decoding and the second stage of two-stage decoding. A classifier used in the first stage of the two-stage decoding is trained only with word n -gram features.

Chapter 6

Experiment

In this section, we evaluate our method of collective sentiment classification on four real-world review datasets with user and/or product information. Three of them are existing datasets [PL04, BDP07, MDP⁺11] and the other is newly collected. Accuracy is the most important evaluation we concern.

6.1 Setting

Each document in the datasets is preprocessed by sentence boundary detection and tokenization. Following Pang *et al.* (2002), we induce word unigrams and bigrams as local features. We ignored n -grams that appeared less than a predefined times (we set this number to be six) in the training data to limit the feature size.

We used an online linear classifier called confidence-weighted [DCP08] in our methods.¹ We should emphasize here that the confidence-weighted algorithm is reported to perform as well as SVM in a document-level sentiment classification [DCP08] and it thereby constructs a strong baseline.

¹The code was kindly provided by the author of the paper.

For each confidence-weighted classifier, there are two hyper-parameters (confidence parameter ϕ and the number of iterations for training) that need to be defined. Confidence-weighted learning adjusts a multivariate Gaussian distribution over the weight parameters where ϕ controls the update rate of the variance and the mean. Given a larger ϕ , the variance would decrease faster and the mean would be updated more gradually. The number of iterations controls how many times each training instance is used to update the parameters. We set the ϕ to be 1.0 and set iteration to be 10.

6.2 Datasets

6.2.1 A New Dataset: IMDB

Our proposals take advantages of large-scale datasets since they provide more global features while there are more reviews belong to one user or one product. Existing datasets focus on the textual based classification thus they are not perfectly suitable. We want each user and each product to have as many reviews collected as possible such that there will be more available global features. Instead of crawling random product reviews, we target on crawling movie reviews on Amazon.com because the movie reviews are consider difficult for classifier to estimate a correct label[ZJZ06].

The crawling is performed in a breadth-first way that for each task it collect all the reviews before adding new tasks. The task here is a movie or a user. For each task, it saves all the reviews and put unseen movies or users into task queue. We choose 10 top ratted movies and 10 bottom ratted movies as our seeds.

The features of the collected review contains: content, user id, movie id, rating ranging from 1 to 10, title, time and the usefulness. In our experiment,

Dataset	Pang	Blitzer	Maas	IMDB
No. of reviews	2000	188,350	50,000	1,001,240
No. of users	309	123,584	n/a	351,392
No. of products	1107	101,021	7,036	70,776
No. of reviews/user	6.5	1.5	n/a	2.8
No. of reviews/products	1.8	1.9	7.1	14.1

Table 6.1: Dataset statistics.

only content, user id, movie id and rating will be considered.

6.2.2 Details

Pang *et al.* (2004), Blitzer *et al.* (2007) and Maas *et al.* (2011) collected three datasets that contain user and/or product information. All of the polarities (positive and negative labels) in these datasets are balanced. Our newly collected dataset is also balanced according to positive and negative labels. Table 6.1 summarizes the statistics of four datasets.

Pang: This dataset is a small subset of reviews manually chosen from a movie review archive.² The archive is collected from a discussion newsgroup on art movies. 2,000 reviews are randomly picked from a large archive which contains over 30,000 reviews.

Blitzer: This dataset is collected from a shopping website³ on various domains of products. We used part of its total 780k reviews to be consistent with the other two datasets. We automatically delete replicated reviews

²<http://reviews.imdb.com/Reviews>

³<http://www.amazon.com>

written by the same author on the same product (resulting in 740k raw reviews). Then the reviews are balanced for positive and negative labels (over 90k reviews for each, by randomly sampling the same number of positive reviews as the negative reviews).

Maas: This dataset is collected also from the same movie review website as the Pang dataset except that the reviews are not constrained on any discussion newsgroup. The picking process is automatically performed by collecting (upper-bounded number of) reviews for each product (movie).

IMDB: This is our newly collected dataset from IMDB. Each document in this dataset is a movie review. Different from the Pang dataset, the domain of our dataset is not limited to a discussion newsgroup. After balancing the positive and negative labels, there are about one million review left.

We automatically recovered the user and product information (implicitly) included in the datasets. The Pang and Blitzer datasets are accompanied with the original *html* files, from which we automatically extracted the user and product for each review. We used a URL (link to the movie title) provided by the Maas dataset for each review as the identifier of a product, a movie in this case. Because user information cannot be fully recovered in the Maas dataset, we only consider the product popularity on this dataset.

When we split the datasets for cross-validation, we maintained the order of the Pang dataset and shuffled the Blitzer, Maas and IMDB datasets for training and testing before splitting. Since our method takes advantage of the user and product information, the more reviews each user or product has, the higher accuracy our method is expected to achieve. In the Blitzer and Maas datasets, the reviews were originally ordered by user and product, respectively. Then, if we naively use the original order given by the

datasets without shuffling in splitting them, the average number of reviews for each user or product becomes unnaturally high, in which our methods would unfairly take advantages⁴. In order to prevent the seemingly unfair accuracy gain under this particular splitting, we shuffled the reviews before any experiment rather than using the split provided by the authors. On all of the three datasets, we performed a 2-fold cross-validation.

6.3 Result

6.3.1 Overall Accuracy

We compared the accuracy of our methods with two methods: a baseline method used confidence-weighted linear classifier with n -gram features and an existing user-aware sentiment classifier proposed by Seroussi *et al.* (2010). For reference, we also listed the results reported in Maas *et al.* (2011), which was evaluated using a different 2-fold splitting.

Seroussi *et al.* (2010) proposed a framework that combines scores given by classifiers trained on other users according to the similarity to the target user. We build a personalized classifier for each user on her/his training reviews if she/he has more (positive and negative) reviews than a predefined threshold. For any pair of the users, they compute the similarity as the jacquard distance of word n -grams from their (testing and training) reviews (called “AIT”, which performed best in their paper). To classify a review written by a given user, they combine the scores generated by the other users’ personalized classifiers weighted by the similarities between those users and the given user. We set the aforementioned threshold to be six. In our datasets, many test users had the similarity of 0 to the users in the training

⁴This assumption has been confirmed by experiments.

Method	Pang	Blitzer	Maas	IMDB
Seroussi <i>et al.</i> (2010)	78.05	89.33	n/a	n/a
Maas <i>et al.</i> (2011)	88.90	n/a	88.89	n/a
baseline	86.00	90.10	91.49	87.42
proposed (bias)	86.15	91.16 ≫	92.65	87.57≫
proposed (correlation)	85.95	90.18≫	n/a	88.09≫
proposed (bias+correlation)	86.00	91.16 ≫	n/a	88.15 ≫

Table 6.2: Accuracy (%) on review datasets. *bias* and *correlation* mean using the corresponding global features defined in chapter 3 and 4. Accuracy marked with “≫” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).

data because the size of reviews written by each user is much smaller than that in the Seroussi’s dataset. For labeling the reviews written by such test users, we constructed and used a default classifier trained on all the training data. This approach cannot handle Maas dataset because there is no user information given in this dataset. Neither can it run on IMDB dataset because computing the similarity between users consumes quadratic time complexity as the number of users.

The detailed settings of bias features, cluster parameters for correlation features and decoding strategy are decided according to the experiment of Section 6.3.2, 6.3.3 and 6.3.4, respectively. The decoding strategy we used here are two-stage for IMDB dataset and easiest-first for the rest three datasets since easiest-first is infeasible on IMDB dataset.

Table 6.2 shows the experimental results. The “bias” proposal and “correlation” proposal are identical to the two methods we discussed in chapter 3

and 4. The “bias+correlation” is a combination of the two kinds of global features. Our method improved accuracies on all the four datasets against the baseline classifier. More global features brings higher improvement.

On the Pang dataset, proposed methods contribute little to the accuracy probably because the size of this dataset is small. We will further analyse the reason in Chapter 7. Correlation features relatively have greater impact than bias features. On the Blitzer and IMDB dataset, both proposed global features improve the accuracy with significance. Correlation features performs worse on Blitzer dataset but better on IMDB dataset. On the Maas dataset, by simply using one kind of bias features (the product popularity global features) our method gained the greatest margin of improvement over all the four datasets.

Seroussi *et al.* (2011) performed badly because the reviews size for each user in our datasets is lower than theirs. Then, the personalised classifiers learned on limited instances would be unreliable.

6.3.2 Accuracy of Bias Features

In this section, we compare our polarity bias to the baseline method. The settings are similar as in Section 6.1 except the global features. We here test the improvement by individually adding the user, target and both user and target global features. The decoding strategy for the three smaller datasets is easiest-first to obtain a higher accuracy. For the newly collected IMDB dataset, we pick the two-stage decoding because easiest-first is infeasible. The target global features cannot be applied for Maas dataset because target information is not available.

Proposed polarity bias features generally improves accuracy compare to baseline method. Adding both bias features (user or target based) almost

Method	Pang	Blitzer	Maas	IMDB
baseline	86.00	90.10	91.49	87.42
<i>+user</i>	86.05	91.05 \gg	n/a	87.47 $>$
<i>+target</i>	86.15	90.19	92.65	87.50
<i>+user+target</i>	86.10	91.16\gg	n/a	87.57\gg

Table 6.3: Accuracy (%) on review datasets. *+user* and *+target* mean using the corresponding global features defined as polarity bias in chapter 3. Accuracy marked with “ $>$ ” and “ \gg ” was significantly better than baseline ($p < 0.05$ and $p < 0.01$ assessed by McNemar’s test).

always performs better than adding single one.

Among all the datasets, accuracy improvement on Blitzer and IMDB dataset are more significant than the others. User leniency features are more useful than product popularity features on these two datasets. On the Pang dataset, little improvement has been acquired, which will be analysed in detail in next chapter.

6.3.3 Accuracy of Clusters for Correlation Features

In this section, we evaluate the impact of cluster number for correlation features as denoted in Section 4.3. The cluster number, as a major hyper-parameter k of a clusterer, controls the effectiveness of our global features. We here try to explore the best settings of the hyper-parameters according to the accuracy our classifier performs.

The setting in this experiment remains the same with the setting introduced in Section 6.1 with some exceptions. The datasets we here tested on are only three with both user and target information available, namely Pang,

		User cluster k				
		5	10	20	40	n/a
Target cluster k	5	86.15	86.15	86.15	86.15	86.15
	10	86.25	86.25	86.25	86.25	86.25
	20	86.40	86.40	86.40	86.40	86.40
	40	86.10	86.10	86.10	86.10	86.10
	n/a	85.75	85.75	85.75	85.90	85.95

Table 6.4: Accuracy (%) on dataset **Pang** when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.

Blitzer and IMDB dataset. As two-stage performs very fast decoding with comparative accuracy⁵, we run all the test using this decoding strategy.

We used a opensource k -means implementation⁶ to perform clustering. For each user or target, we collect all the reviews (training and testing) that written by the user or on the target for extracting features. After collecting the reviews, unigram features of the reviews are extracted to represent the user or the target. By setting different hyper-parameter k of the cluster, we have different granularities for users and targets. We test the accuracy when setting different k for users and targets.

As illustrated in Table 6.4, 6.5 and 6.6, clustering almost always improves the accuracy on all the three datasets. The k parameter for users and targets, which gives the highest accuracy, has a weak relation with the actual number of users and targets. Such that, we choose a different cluster choices for every datasets. Generally, the more users and targets there are, bigger k value

⁵Given in Section 7.1.

⁶<http://www.tkl.iis.u-tokyo.ac.jp/~ynaga/yakmo/>

		User cluster k				
		100	200	400	800	n/a
Target cluster k	100	91.21	91.22	91.21	91.19	91.13
	200	91.19	91.19	91.18	91.16	91.09
	400	91.16	91.16	91.15	91.12	91.07
	800	91.13	91.13	91.13	91.10	91.01
	n/a	90.37	90.35	90.34	90.31	90.18

Table 6.5: Accuracy (%) on dataset **Blitzer** when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.

should be set.

On the Pang dataset, as shown in Table 6.4, best accuracy appears when setting k of target to be around 20. User clustering has no effect or negative effect on this dataset (by referring to the row target cluster $k = n/a$ in Table 6.4). This means the users in Pang dataset has limited mutual similarity but the targets could be grouped to decrease the sparsity of correlation global features.

On the Blitzer dataset, as shown in Table 6.5, clustering for both users and targets greatly reflects higher accuracy. Compared with user cluster, target cluster generally increase the accuracy more. The best accuracy appeared when setting very small k for both users and targets. The first reason for this is that there are such many users (123,584) and targets (101,021) that rarely could correlation global features be discovered under such situation (sparsity problem). Another reason is that the name of the product are scattered even by a minor difference. For example, “iPhone black” and “iPhone white” are treated different products while they are extremely similar.

		User cluster k				
		250	500	1000	2000	n/a
Target cluster k	30	87.76	87.83	87.89	87.75	88.09
	60	87.73	87.80	87.87	87.74	88.03
	125	87.69	87.76	87.82	87.68	87.95
	250	87.65	87.71	87.73	87.64	87.85
	500	87.61	87.64	87.65	87.54	87.74
	1000	87.47	87.49	87.49	87.38	87.67
	n/a	86.83	87.08	87.28	87.28	87.59

Table 6.6: Accuracy (%) on dataset **IMDB** when varying the cluster parameter k for users and targets. n/a denote no cluster is performed.

On the IMDB dataset, as shown in Table 6.6, a higher accuracy is obtained when setting k for targets small and k for users large. The targets, movies, are mostly closely related to each other. This is because movies are normally divided by their genres (e.g. “horror”, “scientific” or “love”) directors or actors and actress, for which comments tend to be consistent within one division. For example, users who love movie *The Shawshank’s Redemption* would also love movie *City of God* because they both are in the genres of “*Crime*” and “*Drama*”. Clustering such movies together effectively solve the sparsity problem of global features. On the other hand, users in this dataset tends to more diverged. So the cluster parameter k should be set larger to tolerant the diversity⁷.

⁷We should set higher k for users in this dataset. However due to the computing power the clustering is not feasible in a reasonable time.

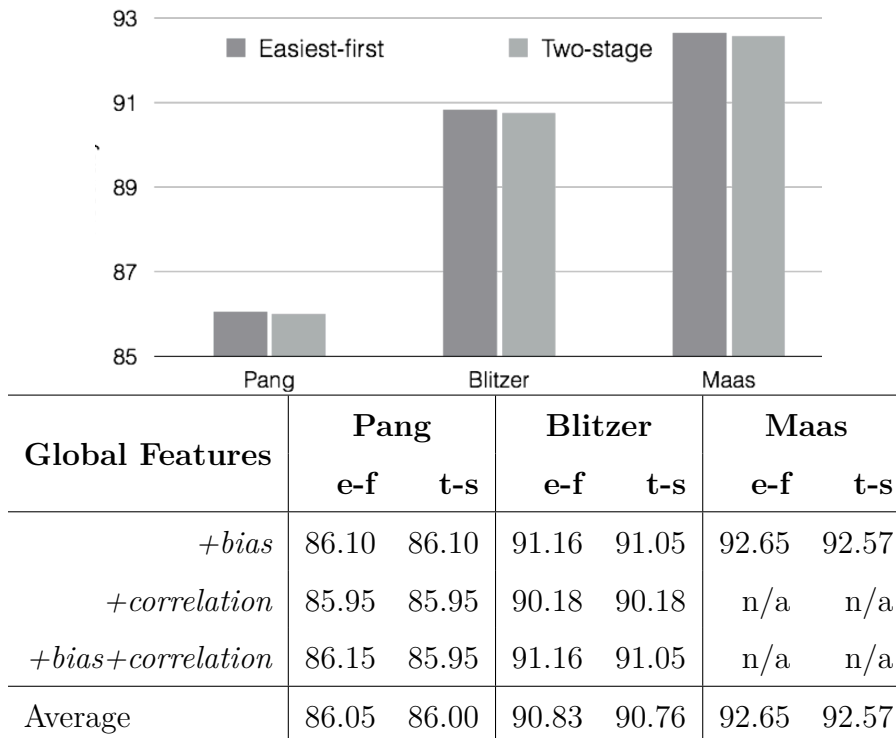


Table 6.7: Accuracy (%) on three datasets when choosing different decoding strategies, namely easiest-first (e-f) and two-stage (t-s). *+bias* and *+correlation* mean using the corresponding global features defined as polarity bias in Chapter 3 and polarity correlation in Chapter 4.

6.3.4 Accuracy of Decoding Strategy

In this section, we investigate the performance of the two decoding strategies introduced in Chapter 5 in terms of accuracy. All the setting remains the same as introduced in Section 6.1 except that the decoding strategy differs as a hyper-parameter. The newly collected IMDB dataset is not used in this test because easiest-first decoding is infeasible for such huge number of test reviews.

As shown in Table 6.7, easiest-first almost always has a minor advantage

over two-stage strategy in the table. The easiest-first takes a more careful process for computing the global features. Thus who simply pursuing a higher accuracy, show deploy easiest-first decoding over two-stage decoding.

Chapter 7

Analysis

In this chapter, we analyze the reasons of our experimental result on accuracy. There mainly three parts in this chapter, namely analysis for polarity bias, analysis for polarity correlation and analysis for decoding strategy.

We first investigate the speed and accuracy of the two decoding strategies when we change the testing data size in Section 7.1. The easiest-first decoding strategy gives a clear advantage in accuracy under the same condition as the two-stage decoding. However, easiest-first consumes theoretically more time on computing. In this section, we will statistically analyse the time consumption and the accuracy improvement.

Polarity bias proposal is then assessed under different standards and settings in Section 7.2. These assessments provide insight into the reasons why this proposal work and under which condition the proposal would further improve.

At last, we analysis the examples of polarity correlations. We try to show which kind of rule could be learned by setting the new features.

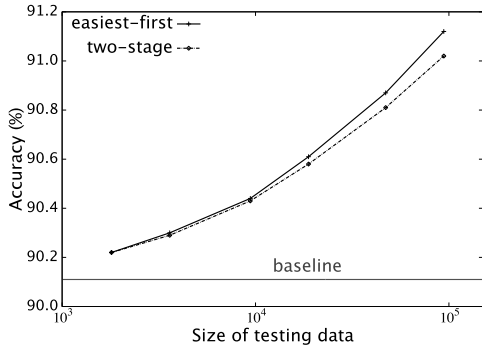


Figure 7.1: The classification accuracy computed accumulatively when we changed the size of testing reviews.

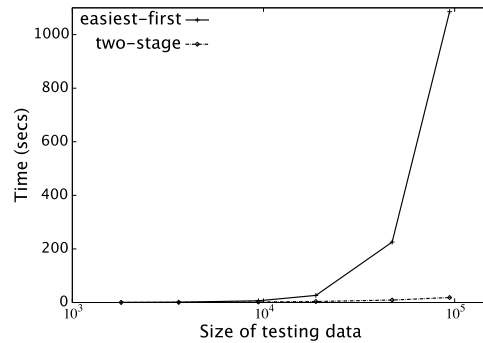


Figure 7.2: Average computation time when we changed the size of testing reviews.

7.1 Speed and Accuracy of Decoding Strategy in Terms of Data Size

First, we investigate the impact of the number of test reviews on speed and accuracy in our collective sentiment classification. We use the Blitzer dataset for evaluation because of its larger size. The user leniency and product popularity are both considered. We use the fixed hyper-parameters ($\phi = 1.0$, $\# \text{ iterations} = 10$) for all the confidence-weighted classifiers used in this experiment.

To illustrate the impact of the size of test data on classification accuracy, we changed the size of test reviews processed at once. Here, instead of decoding on the whole testing data, we split the test reviews into equally-sized smaller subsets and applied our classifier independently to each of the subsets.¹ We accumulate the results for all the subsets to compare the accuracy for the entire test data. Fig. 7.1 shows the experimental results. When

¹We used the same 2-fold cross-validation as the main experiment of accuracy.

		No. of product-related neighbors ($ \mathcal{S}_p(r) $)			
		0	1	2	3-
No. of user-related neighbors ($ \mathcal{S}_u(r) $)	0	120 80.83 (+0.00)	52 82.69 (+0.00)	23 82.61 (-4.35)	27 96.30 (+0.00)
	1	41 85.37 (+2.44)	32 90.63 (+3.13)	12 58.33 (-16.66)	15 100.00 (+0.00)
	2	48 85.42 (+0.00)	17 82.35(-5.88)	4 75.00 (+0.00)	9 66.67 (+0.00)
	3-7	201 87.06 (+0.00)	90 82.22 (-1.11)	57 82.46 (+1.75)	54 88.89 (-1.85)
	8-	609 84.73 (-0.66)	299 87.29 (+0.67)	138 87.68 (-2.17)	152 87.50 (-1.32)

Table 7.1: Accuracy (% , lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the **Pang** dataset divided according to the number of reviews written by the user and the number of reviews on the product. The float inside parentheses is the difference from the baseline method. No accuracy is significantly different from baseline ($p \geq 0.01$ assessed by McNemar’s test).

we process a larger number of reviews at once, we have more reviews per user or per product to compute the global features. The computed global features thereby become more statistically reliable and then accurately capture the user leniency and product popularity, which resulted in the higher classification accuracy. We will confirm this in Section 7.2.1.

We then measured the testing speed by using the same setting as the above experiment while evaluating the average time consumed by one single subset. As shown in Fig. 7.2, the speed of the easiest-first decoding drastically slows down as the number of processed reviews grows, whereas the speed of

		No. of product-related neighbors ($ \mathcal{S}_p(r) $)			
		0	1	2	3-
No. of user-related neighbors ($ \mathcal{S}_u(r) $)	0	55,043 90.12 (+0.01)»	34,735 90.13 (+0.27)»	16,601 90.90 (+0.67)»	9,630 92.37 (+0.56)»
	1	10,768 91.14 (+1.33)	6,530 91.33 (+2.07)	2,974 91.36 (+1.34)	1,536 92.25 (+1.04)
	2	4,595 91.80 (+2.13)»	2,711 91.26(+2.73)	1,292 90.48 (+1.63)	663 91.98 (+2.04)
	3-7	8,120 92.45 (+2.35)»	4,974 91.19 (+2.37)»	2,174 92.23 (+3.50)	998 89.98 (+1.70)
	8-	13,243 93.66 (+1.94)»	7,484 92.34 (+1.80)»	3,017 91.32 (+1.46)»	1,289 90.07 (+1.55)

Table 7.2: Accuracy (% , lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the **Blitzer** dataset divided according to the number of reviews written by the user and the number of reviews on the product. The float inside parentheses is the difference from the baseline method. Accuracy marked with “»” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).

the two-stage decoding increases linearly. Meanwhile, the accuracy of the two strategies are competitive as shown in Fig. 7.1.

Based on these observations, the key factor in achieving the better accuracy is not the choice of decoding strategy but the size of test data processed at once. We conclude that when we have too many test data for the easiest-first decoding to process in a practical time, we should adopt two-stage decoding strategy to induce and exploit more reliable global features. Otherwise, we can choose the easiest-first decoding to enjoy a modest gain in accuracy.

No. of product-related neighbors ($ \mathcal{S}_p(r) $)				
0	1	2-5	6-10	11-
3,597	4,646	14,394	10,444	16,919
86.41(+0.27) \gg	91.05(+2.00) \gg	92.48(+1.55) \gg	93.96(+1.22)	93.69(+0.74) \gg

Table 7.3: Accuracy (% , lower inside cell) of proposed method (two-stage) and the review size (upper inside cell) on the **Maas** dataset divided according to the number of reviews on the product. The float inside parentheses is the difference from the baseline method. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).

7.2 Analysis for Polarity Bias

In this section, our polarity bias proposal is discussed in mainly three aspects. First, we analyse the properties of the datasets under the criterion of accuracy, namely the size of neighbours (Section 7.2.1) and the bias distribution (Section 7.2.2). Then, we show the impact of the training data by measuring the accuracy of unknown users and targets (Section 7.2.3) and the accuracy when giving different size of training data (Section 7.2.4). At last, we show some examples in Section 7.2.5.

This section tests three existing datasets, Pang, Blitzer and Maas. Only the polarity bias proposal is discussed in this section.

7.2.1 Accuracy in terms of neighbor size

The accuracy gain is rooted in the global features while the global features are computed by referring to labels of the (user- and product-related) neighboring reviews, $\mathcal{S}_u(r)$ and $\mathcal{S}_p(r)$. When only one of such neighbors is

available, the global features may be unreliable compared to those computed from many neighbors. We then investigate how the number of user- and product-related neighboring reviews would affect the accuracy improvement.

Table 7.2 shows that the user leniency features greatly contribute to the improvement and the product popularity has limited influence on the Blitzer dataset. The popularity features play an important role on the Maas dataset as shown in Table 7.3. Both user leniency and product popularity features show no improvement on the Pang dataset as in Table 7.1 because of the limited review size as we illustrated in Section 3.2 (where our method contributes less improvement in small size of test data). It is also probability because the biases of user and product is not enough in this dataset, which we will show in Section 7.2.2. In general, we can expect further improvement if we collect some unlabeled reviews for the user (or product).

We noticed that when the number of reviews written by a user or on a product is large enough ($3 \leq |\mathcal{S}_u(r)| \leq 7$ in the Blitzer dataset and $2 \leq |\mathcal{S}_p(r)| \leq 5$ in the Maas dataset), having more reviews for such users and products does not improve the accuracy any further. Considering that larger $|\mathcal{S}_u(r)|$ or $|\mathcal{S}_p(r)|$ results in lower speed of easiest-first decoding as the time complexities we analyzed in Section 5.2 and the speed shown in Fig. 7.2, we could bound the number of reviews written by each user or on each product to save computation without losing accuracy.

7.2.2 Polarity bias in terms of user or product

Since our method takes advantage of the biased distributions over polarity labels in terms of a user or a product, the larger the bias exists in the data the greater our method could improve the classification accuracy. We then compute the polarity bias in terms of users (in short, *user_bias*) and

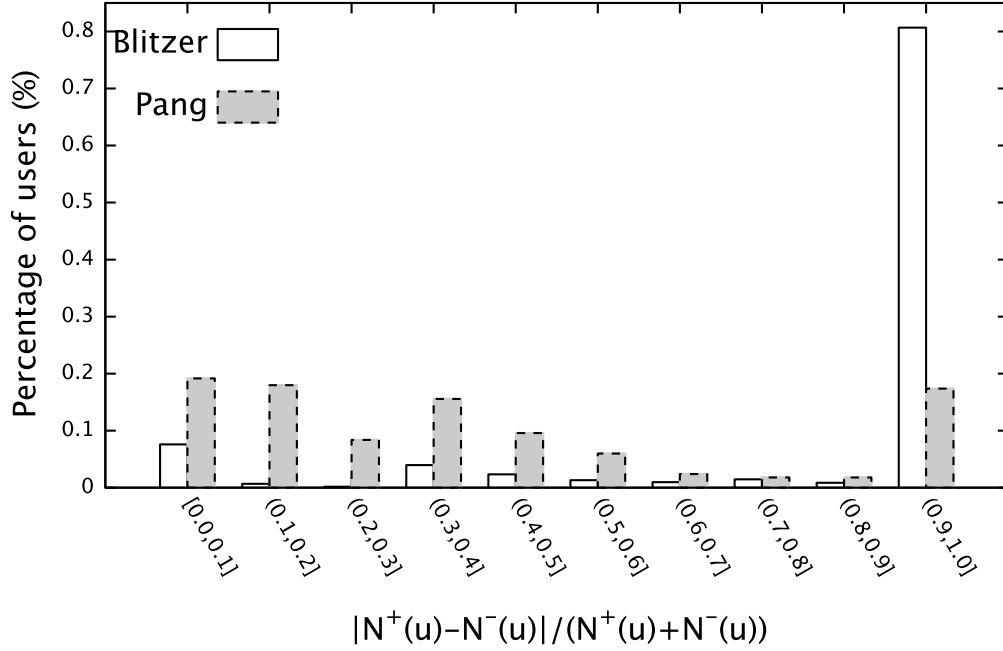


Figure 7.3: The user bias distribution. The users who only have one review are eliminated.

products (*product_bias*) as follows:

$$user_bias(u) = \frac{|N^+(u) - N^-(u)|}{N^+(u) + N^-(u)},$$

$$product_bias(p) = \frac{|N^+(p) - N^-(p)|}{N^+(p) + N^-(p)},$$

where, $N^{+/-}(u)$ or $N^{+/-}(p)$ are the number of reviews written by user u or written on product p with polarity $\in \{+, -\}$.² With this definition, for instance, user u who only writes positive reviews will have $user_bias(u) = 1$, while user u who writes positive and negative reviews evenly will be assigned as $user_bias(u) = 0$. The higher $user_bias$ and $product_bias$ values exist, the more potential our method could improve the accuracy.

²The numbers are counted using all the reviews, including training and testing data.

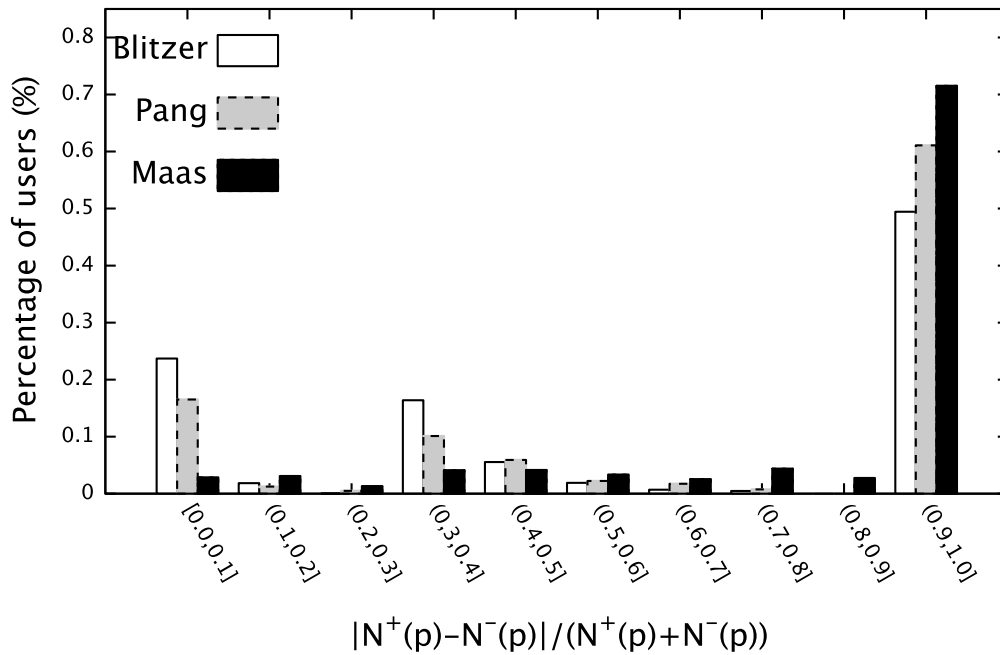


Figure 7.4: The product bias distribution. The products who only have one review are eliminated.

Due to the different collecting methods, the three datasets we used show different bias properties. The distributions of *user_bias* and *product_bias* values are shown in Fig. 7.3 and Fig. 7.4 respectively. Maybe because the Pang dataset is collected from a discussion newsgroup, the users in it are less biased than those in the Blitzzer which is collected from a general domain. Such that, the user leniency features extracted from the Pang dataset might be unreliable as users do not have much of bias. The products in the Maas dataset are more biased than the other two datasets. This could be a reason why user information in the Blitzzer dataset and product information in the Maas dataset contribute the most to the improvement.

As illustrated in Fig. 7.1, when the size of reviews is small, the accuracy

	(su, sp)	(uu, sp)	(su, up)	(uu, up)	total
# reviews	951	86	850	113	2000
# reviews/user	3.98	1.13	3.85	1.18	4.41
# reviews/product	1.69	1.08	1.16	1.02	1.44
baseline	87.38	88.37	84.71	82.30	86.00
our (easiest-first)	87.07	87.21	84.24	81.42	85.55
our (two-stage)	87.07	87.21	84.12	81.42	85.50

Table 7.4: Accuracy (%) on seen/unseen user or product splits of the **Pang** dataset. *su*, *uu*, *sp* and *up* stand for *seen user*, *unseen user*, *seen product* and *unseen product* respectively. No accuracy is significantly different from baseline ($p \geq 0.01$ assessed by McNemar’s test).

of our method decreases. Then, the small size of Pang dataset (Table 6.1) and the low user bias values are seemingly to be the reasons why our method performed badly on the Pang dataset.

7.2.3 Impact of training reviews written by test users (or on test products)

Aiming at revealing how well our method works for reviews written by unseen (emerging) users or on unseen (emerging) products, we investigated the classification accuracy in terms of whether we have observed the same user (or product) in the training data or not. We use user leniency and product popularity features on the Pang and Blitzer datasets, while we consider only product popularity features on the Maas dataset. The baseline classifier is expected to better estimate the labels of reviews written by seen users or on

	(su, sp)	(uu, sp)	(su, up)	(uu, up)	total
# reviews	35,689	60,775	36,859	55,027	188,350
# reviews/user	2.04	1.04	2.14	1.04	1.40
# reviews/product	1.20	1.39	1.14	1.20	1.43
baseline	89.72	90.33	90.50	89.94	90.14
our (easiest-first)	91.39>>	90.91>>	92.34>>	90.33>>	91.11>>
our (two-stage)	91.24>>	90.87>>	92.13>>	90.29>>	91.02>>

Table 7.5: Accuracy (%) on seen/unseen user or product splits of the **Blitzer** dataset. *su*, *uu*, *sp* and *up* stand for *seen user*, *unseen user*, *seen product* and *unseen product* respectively. Accuracy marked with “>>” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).

seen products than those unseen ones’, because the classifier learns n -grams specific to these users or the products (and thus effective in classification). On the other hand, our method performed well when more reviews are available for users and products in the test data, so we can expect consistent improvement for both seen and unseen users (products).

On the Maas dataset as shown in Table 7.6, the improvement on the reviews written on unseen products is significantly larger than the reviews on seen products. This may seem counterintuitive, since we have a smaller number of reviews written on the unseen products (which means less reliable global features). The reason is probably that the baseline classifier performed poorly on the reviews written on unseen products, and hence left our method larger space for improvement.

As shown in Table 7.5, a larger improvement was observed on reviews written by the seen users in the Blitzer dataset. We found that the average

	(sp)	(up)	total
# reviews	46,397	3,603	50,000
# reviews/product	4.82	1.62	4.22
baseline	91.93	85.62	91.47
our (easiest-first)	93.07	87.73 \gg	92.68
our (two-stage)	93.02	87.59 \gg	92.63

Table 7.6: Accuracy (%) on seen/unseen product splits of the **Maas** dataset. *sp* and *up* stand for *seen product* and *unseen product*. Accuracy marked with “ \gg ” was significantly better than baseline ($p < 0.01$ assessed by McNemar’s test).

number of reviews written by a user was extremely low (1.04 reviews) and no global features were fired in most of these reviews. We consider this may be the main reason for the poor improvement of accuracy on reviews written by the unseen users.

On the Pang dataset as shown in Table 7.4, the smaller number of training data resulted in poor classification accuracy especially on the unseen users and unseen products. Unlike on the other two datasets, the lack of biases and the small number of reviews seem to be the reason to blame.

7.2.4 Learning curves

Using the same setting as the analysis in Section 7.1, we divided the training data and investigated the effect on accuracy. Fig. 7.5 shows the accuracy when we change the size of training data. Our method had a clear advantage over the baseline method even when the size of training data is small (1,800 reviews). In other words, we don’t need much training data to

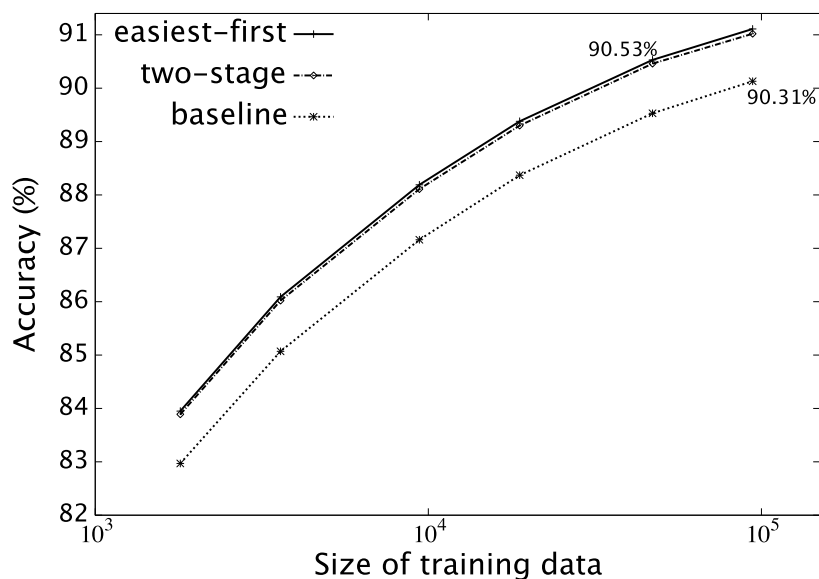


Figure 7.5: Average accuracy when we change the size of training data.

learn the correlation between the label and the global features. Our method trained with half of the training data achieved a higher accuracy (90.53%) than the baseline method trained on the entire data (90.31%).

7.2.5 Examples

Some examples are given here to explain how our model works. As shown in Table 7.7, our method successfully classifies some reviews that are hard to be correctly classified when only textual features are used.

In the first two examples, weak negative textual features are found in the test review. However, since the two users are lenient and the product of the first review is relatively popular (these characteristics are captured by our proposed method), these two reviews should still be given positive labels.

Frequently, sentiment expressed inside a review is not obvious if the classifier does not know the meaning of the words (sometimes, even a human

user len	target pop	review	labels gold bl our
$f_u^+ : 0.92$ $f_u^- : 0.08$	$f_p^+ : 0.67$ $f_p^- : 0.33$... The book would deserve 5 stars if the author had compared several jurisdictions popular instead of focusing solely on Nevada.	+ - +
$f_u^+ : 0.81$ $f_u^- : 0.19$	$f_p^+ : 0.50$ $f_p^- : 0.50$...I am using Windows XP with office Pro 2003 and today was disappointed to find that the Help menu is not as user friendly or helpful as earlier editions	+ - +
$f_u^+ : 0.18$ $f_u^- : 0.82$	$f_p^+ : 0.00$ $f_p^- : 1.00$	ooo! see Halle act. act, halle, act. emote. emote. see halle act drunk. see halle act crying. see halle act nympho. ... but what does it matter, since we get to see halle act ...	- + -

Table 7.7: Examples show the influence of leniency and popularity global features. The **bold** content is the negative evidence learned by classifier. bl is baseline method.

Rank	Feature	Weight
1	Lars Lindahl ⁺ _Chuck Dowling	0.391
2	Berge Garabedian ⁻ _Jamey Hughton	-0.342
3	E. Benjamin Kelsey ⁻ _Andrew Cunningham	0.325
4	Eugene Novikov ⁻ _Nathaniel R. Atcheson	-0.319
5	Chuck Dowling ⁺ _Lars Lindahl	0.319
6	Dustin Putman ⁻ _Joe Chamberlain	0.312
7	Andrew Hicks ⁻ _Larry Mcgillicuddy	0.308
8	Jamey Hughton ⁺ _Robert Workman	0.301
9	Robert Workman ⁺ _Jamey Hughton	0.295
10	John Sylva ⁺ _Tim Voon	0.291

Table 7.8: The top ten correlation user features ($u_i^{y_i} - u_j$) in **Pang** dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).

feels hard to identify sentiment from these words). As we can see in the third example in Table 7.7, the baseline classifier could recognize no obvious sentiment evidence from the textual features, while our method classified it as negative by detecting that its on a notorious product and the user is critical.

These examples illustrate that our model can successfully exploit the user and product biases to improve the accuracy of sentiment classification.

7.3 Correlation Examples

This section illustrates some examples that our correlation proposal gives. Under the framework of supervised classification, the weights of correlational

Rank	Feature	Weight
1	The Phantom ⁻ _The Matchmaker	0.308
2	Nowhere ⁻ _The Matchmaker	0.308
3	Lucas ⁺ _The Matchmaker	0.308
4	Soul Food ⁺ _The Matchmaker	0.308
5	The Arrival ⁺ _The Matchmaker	0.308
6	Analyze This ⁺ _American Pie	0.303
7	200 Cigarettes ⁻ _The 13th Warrior	-0.295
8	American Pie ⁺ _Analyze This	0.293
9	U-571 ⁺ _Frequency	0.291
10	Gladiator ⁺ _Frequency	0.291

Table 7.9: The top ten correlation target features ($t_i^{y_i} - t_j$) in **Pang** dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).

features are learned in training data (only one fold is used here). Larger value (positive or negative) the weight is, more significant is the corresponding feature for estimating a label (Equation 2.4).

Besides listing the most significant correlation features of each dataset, we analysis the targets and the users contained in the features accordingly. For users, we analysis the characteristics of them by available information. For movies or products, we show the properties that given by the corresponding website. For clusters, we show the properties that shared by the important members of the clusters.

Rank	Feature	Weight
1	Birchbunch5 ⁺ _T. Cochran	-4.559
2	Bob Olson ⁺ _Lorraine A. Willis	-4.549
3	Momof2girls Pat ⁺ _Tomarah Hutton	-4.320
4	Matthieu P. Raillard ⁺ _K. B. Cournand	-2.834
5	K. Fontenot ⁺ _Gordon Ball	-2.605
6	Chris Taylor ⁻ _Damien Desa	-2.440
7	Lee ⁺ _The drew	-2.407
8	P. Demers ⁻ _D. Mulroy	-2.357
9	Jemimagold ⁻ _Sydzack	-2.091
10	F. Buchanan ⁻ _Samtwin	-1.977

Table 7.10: The top ten correlation user features ($u_i^{y_i} - u_j$) in **Blitzer** dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).

7.3.1 None Clustering

In this section, users and targets are directly referred by the names ($k = n/a$ in Section 6.3.3). Compared with clustered targets and users, no clustering setting is more intuitive when showing the result. After training the classifier, for each dataset we sort the absolute values of weights ($|w_i|$, where \mathbf{w} is the weight vector in Equation 2.4) for our correlation global features and list the highest 5 user correlation features ($u_i^{+/-} - u_j$) and 5 target correlation features ($t_i^{+/-} - t_j$).

We mainly focus on two phenomenas in this section. First, if a user or target frequently appears in the lhs of the correlation features, the user must writes vague text or the target must written by vague text so the classifier needs other features (correlation features in this case) to estimate correctly.

Rank	Feature	Weight
1	How To Be A People Magnet (book) ⁻ ₋	-3.777
	To Thine Own Self Be True (book)	
2	The Secret World Of Mermaids (book) ⁻ ₋	-2.0388
	Fairy Land (book)	
3	BFW01 Laptop Speaker ⁺ ₋ Orthopaedic... (book)	1.913
4	Marry Poppins (book) ⁻ ₋	-1.859
	Life Application Study... (book)	
5	24 Hours Party People (DVD) ⁻ ₋	-1.859
	Life Application Study... (book)	
6	M084100 Vacuum Cleaner ⁺ ₋ Quiet Strength (book)	-1.726
7	Memorex CD/DVD Labels... ⁺ ₋	1.604
	The legend of zelda...(toy)	
8	Eerth Wind & Fire (CD) ⁺ ₋ The Legend Of Zelda...(toy)	1.604
9	The Bluegrass... (book) ⁻ ₋	-1.565
	Image transfer on clay (book)	
10	The New Power Soul (CD) ⁻ ₋ Moog Indigo (CD)	1.478

Table 7.11: The top ten correlation target features ($t_i^{y_i} - t_j$) in **Blitzer** dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).

Second, if a user correlation feature or a target correlation feature is ranked higher by the absolute value of weight, the two users or two targets must be highly correlated with a relationship as defined in Section 4.2.

On Pang dataset as shown in Table 7.8 and Table 7.9, the correlation features mostly give positive evidence since most of the weights are positive. By the definition in Section 4.2, user *Lars Lindahl* and user *Chuck Dowling* are

Rank	Feature	Weight
1	gcd70 ⁻ _raggejohansson	-2.412
2	callanvass ⁺ _raggejohansson	-2.412
3	jwiffle ⁺ _jazfingr-1	2.070
4	matowakita ⁻ _jazfingr-1	2.070
5	marcalan-2 ⁺ _HALIFA-le-X	-2.036
6	cineburk ⁻ _dapplegrey13	-1.768
7	yaso28 ⁺ _dapplegrey13	-1.768
8	edwagreen ⁻ _dapplegrey13	-1.768
9	drystyx ⁻ _dapplegrey13	-1.742
10	jbacks3 ⁻ _ragamullin	-1.742

Table 7.12: The top ten correlation user features ($u_i^{y_i} - u_j$) in **IMDB** dataset with highest absolute weight ($w_{u_i^{y_i} - u_j}$).

friends (referring to rank 1 and 5 in Table 7.8) and user *Jamey Hughton* and user *Robert Workman* are friends (referring to rank 8 and 9 in Table 7.8). The movie *The Matchmaker* appears as the second target in correlation features in many our listed top features in Tabel 7.9 (rank 1,2,3,4 and 5). This means the text features that used to describe this movie tend to be ambiguous thus classifier need other supportive evidence to make a correct decision. Movie *American Pi* and *Analyze This* are friends (referring to rank 6 and 8 in Table 7.9).

On Blitzer dataset as shown in Table 7.10 and Table 7.11, the correlation features mostly give negative evidence since most of the weights are negative. The product name is listed with a picture collected from the product's gallery. The correlations occurs mostly between products in the same category (e.g.

Rank	Feature	Weight
1	Prayer of the Rollerboys ⁺ _Fraternity Vacation	-2.411
2	Paul ⁺ _The Blue Bird	-1.768
3	Pi ⁻ _The Blue Bird	-1.768
4	Defiance (I) ⁺ _The Blue Bird	-1.768
5	Zombieland ⁺ _The Blue Bird	-1.768
6	V for Vendetta ⁺ _The Blue Bird	-1.768
7	The White Ribbon ⁻ _The Blue Bird	-1.768
8	Tombstone ⁺ _The Blue Bird	-1.768
9	Halloween: Resurrection ⁻ _Miami Connection	1.732
10	Private Parts ⁺ _Witchouse 3: Demon Fire	1.696

Table 7.13: The top ten correlation target features ($t_i^{y_i} - t_j$) in **IMDB** dataset with highest absolute weight ($w_{t_i^{y_i} - t_j}$).

book category or CD category). In the same category, products with same genre tend to be correlated (e.g. two products in rank 2 of Table 7.11 are both fairytales).

On IMDB dataset as shown in Table 7.12 and Table 7.13, different from pang dataset, the negative instances lack evidence to be correctly classified. As illustrated in Table 7.12, user *raggejohansson* (rank 1 and 2), and user *dapplegrey13* (rank 6,7,8 and 9) write reviews that are difficult to classify by the text. The movie *The Blue Bird (1940)*, shown in Table 7.13, has the same problem that users frequently write confusing comments to it (especially when it is negative).

7.3.2 Clustered Users and Targets

In this section, we discuss the correlation features when targets and users are clustered. We list the ten top features with the highest absolute weight value only for those settings that performed highest accuracy in Section 6.3.3. The statistics shown here are $k = 20$ for target cluster of Pang dataset (Table 7.14), $k = 200$ for user cluster of Blitzer dataset (Table 7.15), $k = 100$ for target cluster of Blitzer dataset (Table 7.16), and $k = 30$ for target cluster of IMDB dataset (Table 7.17).

The statistics are a slightly different than Section 7.3.1. Since the user and target are replaced with the cluster assignments in the correlation features, some most representative users or targets of the cluster are also listed. For each target cluster, we also guessed the genre or category mainly according to the properties listed in product webpage (Amazon or IMDB). The guessed genres are possibly not the true relation.

On Pange dataset, the target correlation features support negative polarity predictions (shown in Table. 7.14). Many interesting phenomena is found, for example people like pure *Romance* movies don't like them mixed with *Comedy* elements (Rank 2 in Table 7.14).

On Blitzer dataset, as shown in Table 7.15 and Table 7.16, most user correlation weights are negative and all target correlation weights are positive. This means users with negative sentiment, e.g. lenient user discussed in Chapter 3, use language with similar patterns which classifier can hardly understand. Similarly, the positive patterns of reviews commented on each product cannot be understood by the classifier. Clustering successfully provide the connectivity by grouping users and targets according to the comments. Some target clusters provide strong positive evidence, for example *c53* in Table 7.16, means that people who like these cluster of products are

more generous on other products.

On Blitzer dataset, as shown in Table 7.15 and Table 7.16, most user correlation weights are negative and all target correlation weights are positive. This means users with negative sentiment, e.g. lenient user discussed in Chapter 3, use language with similar patterns which classifier can hardly understand. Similarly, the positive patterns of reviews commented on each product cannot be understood by the classifier. Clustering successfully provide the connectivity by grouping users and targets according to the comments. Some target clusters provide strong positive evidence, for example *c53* in Table 7.16, means that people who like these cluster of products are more generous on other products.

On IMDB dataset shown in Table 7.17, the correlation features are most helpful to deciding negative instances possibly because users use vague description in their reviews.

Rank	Feature	Weight	Rank	Feature	Weight
1	c15 ⁺ _c7	0.829	6	c7 ⁻ _c12	-0.592
2	c0 ⁺ _c4	-0.805	7	c5 ⁺ _c0	-0.575
3	c3 ⁺ _c11	-0.734	8	c14 ⁻ _c13	-0.563
4	c12 ⁺ _c14	-0.699	9	c11 ⁻ _c2	-0.562
5	c19 ⁻ _c16	0.651	10	c19 ⁻ _c3	0.546

Cluster	Top Movies	Genre
c0	Sexy Beast, Fantasia/2000	Romance
c2	Species II, Varsity Blues	Drama& Sci-Fi
c3	The Right Stuff, Natural Born Killers	Drama&Biography
c4	Pane E Tulipani	Comedy&Romance
c5	Shrek, Krippendorf's Tribe	Comedy
c7	Star Trek: Insurrection, Sphere	Sci-Fi
c11	Star Wars: Episode I, Batman & Robin	Fantasy
c12	The Truman Show	Comedy&Drama
c13	Planet of the Apes	Action&Sci-Fi
c14	Life is Beautiful, The Siege	Drama
c15	Titanic, Starship Troopers	Popular
c16	Deep Impact, 54	Drama
c19	Scream 2, The Big Lebowski	Horror&Crime

Table 7.14: The top ten correlation target (clustered) features ($tc_i^{y_i} - tc_j$) in **Pang** dataset with highest absolute weight ($w_{tc_i^{y_i} - tc_j}$). For each cluster, two most representative movies (separated by comma) and the approximate genre is given.

Rank	Feature	Weight	Rank	Feature	Weight
1	c80+_c1	-4.009	6	c170-_c106	-1.596
2	c45+_c11	-1.839	7	c159+_c179	-1.544
3	c71+_c11	-1.839	8	c106-_c170	-1.430
4	c133+_c186	1.805	9	c187+_c186	-1.401
5	c175+_c179	-1.774	10	c120-_c146	-1.329

Cluster	Top Users
c1	Ernest Jagger, Wiredweird
c11	gregory s. moss
c45	cj, jj
c71	David Ogletree, Barron W. Crist
c80	Rachael D. Roberts
c106	Susan C. Chivers, Mr. Murder
c120	John Matlock, Bruce P. Barten
c133	Deb, Victory Silvers
c146	Andy8047, D. Donovan Editor
c159	Charles J. Rector, Pattipeg S. Harjo
c170	T. Bartlette, Brian A. Quinones
c175	Wonald A. Woodson, vladam@hotmail.com
c179	C. O. Deriemer, Rob Hardy
c186	aj, L. A. Vitale
c187	chandler, steven hancock

Table 7.15: The top ten correlation user (clustered) features ($uc_i^{y_i}_{uc_j}$) in **Blitzer** dataset with highest absolute weight ($w_{uc_i^{y_i}_{uc_j}}$). For each user cluster, two most representative users are listed (separated by comma).

Rank	Feature	Weight	Rank	Feature	Weight
1	c53+_c53	1.900	6	c8+_c53	1.646
2	c18+_c53	1.788	7	c53+_c38	1.615
3	c97+_c53	1.747	8	c81+_c53	1.612
4	c53+_c18	1.734	9	c54+_c54	1.580
5	c45+_c53	1.665	10	c18+_c18	1.559

Cluster	Top Products	Category
c8	Unfaithfully Yours (DVD), Evangelical Feminism and Viblical Truth (book)	Humanity& Comedy
c18	Superman Returns (DVD), Golden Boy (book)	Fantasy
c38	Adam’s Fallacy: A Guide to... (book), On the Trail of the Assassins (book)	Economy& Politics
c45	The Ultimate Fake Book (book), King Kong (music)	Music
c53	Chaos and Creation in the Backyard (music), Darwin’s Black Box (book)	Rock&Science
c54	The Complete Idiot’s Guide to Understand- ing Catholicism (book), Genevieve (book)	Religious& Contemporary
c81	The Resurrection of Christ (book), Into the Sun (DVD)	Religious& Society
c97	Understanding Anti-americanism (book), The Emerging Democratic Majority (book)	Politics

Table 7.16: The top ten correlation target (clustered) features ($tc_i^{y_i}_{tc_j}$) in **Blitzer** dataset with highest absolute weight ($w_{tc_i^{y_i}_{tc_j}}$). For each cluster, two most representative products (separated by comma) and the approximate genre is given.

Rank	Feature	Weight	Rank	Feature	Weight
1	c10 ⁻ _c23	-0.785	6	c21 ⁻ _c9	-0.707
2	c4 ⁻ _c23	-0.784	7	c10 ⁺ _c29	0.689
3	c11 ⁻ _c11	-0.751	8	c21 ⁻ _c17	-0.688
4	c15 ⁻ _c28	-0.739	9	c10 ⁺ _c12	0.687
5	c15 ⁻ _c16	-0.722	10	c12 ⁻ _c29	-0.683

Cluster	Top Movies	Genre
c4	The Fighter (I), On Her Majesty's Secret Service	Bio&Adventure
c9	Casino Royale, The Dark Knight Rises	Action
c10	Iron Man 3, Charlie and The Chocolate Factory	Adventure&Sci-Fi
c11	A Beautiful Mind, True Grit	Bio&Adventure
c12	The Dark Knight	Action&Crime
c15	Goodfellas, The Holiday	Bio&Drama
c16	Anatomy of a Murder, The Runaways	Crime&Drama
c17	Skyfall, Quantum of Solace	Action&Adventure
c21	Star Wars: Episode III, King Kong	Sci-Fi&Adventure
c23	Halloween	Horror
c28	The Spy Who Loved Me, The Man with the Golden Gun	Action&Adventure
c29	Batman, The World Is Not Enough	Action&Fantasy

Table 7.17: The top ten correlation target (clustered) features ($tc_i^{y_i}_{tc_j}$) in **Imdb** dataset with highest absolute weight ($w_{tc_i^{y_i}_{tc_j}}$). For each cluster, two most representative movies (separated by comma) and the approximate genre is given.

Chapter 8

Conclusion and Future Work

In this thesis, we have discussed two proposals that use user and target properties to help sentiment analysis. In this section, we conclude our methods and give future directions of our work.

8.1 Conclusion

Several existing work estimate the sentiment polarity for a given document with consideration of user who wrote the document and the target on which the document is written. However, these approaches are infeasible in some circumstances, especially when user or target is unknown or the training data for such user or target is insufficient.

In this thesis, we have presented two ways of using the user and target to help sentiment classification, namely polarity bias and polarity correlation. They are defined as global features in a supervised classifier that is different from local features because global features cannot be extracted independently from the document.

- In Chapter 3, we discussed polarity bias proposal. We first observed

that user normally gives biased ratings and similarly targets are given biased ratings, too. This bias property is called user leniency and product popularity. For each user and each target we compute two variables, the positive and negative ratio of labels, to describe user leniency and target popularity. The variables is our global features in a supervised classifier.

- In Chapter 4, we discussed polarity correlation proposal. The targets are frequently correlated and the users are also correlated. For each pair of targets and each pair of users, we build two features and let the classifier to learn how much and which direction is the correlation. Furthermore, for resolve the sparsity and unknown user or unknown target problem, clustering is introduced.

In Chapter 5, we discussed two decoding strategy to resolve global dependencies. The newly introduced global features introduced global dependencies between reviews' labels. We first transferred the problem into decoding problem and then we implemented two efficient decoding strategies to solve the problem, name two-stage decoding and easiest-decoding. Compared to easiest-first, time consumed by two-stage decoding is much shorter.

In Chapter 6, we conducted experiments on three existing and one newly collected real-world datasets to compare with the existing approaches. Both our proposed global features outperformed the existing methods on three bigger datasets. The polarity bias features of both user (user leniency) and target (product popularity) outperforms any single one of them. Clustering the users and targets for correlation features with reasonable hyper parameter generally increase accuracy. Easiest-first performs more accurately than two-stage decoding.

In Chapter 7, we analysed our methods by different standards. Easiest-first is proved to be more accurate but consumes much more time than two-stage decoding in practice. The more reviews per user or per target, the larger improvement our method gains. For those unknown users and targets, accuracy is also improved by adding polarity bias features. Some examples of reviews that baseline classifier wrongly estimated while our method correctly estimated are given. At last, the top user correlations and target correlations leaned from training data are listed.

8.2 Future Direction

We consider this method as a first step toward modelling users and targets as global dependencies. In the future, we are going to explore along the following three direction.

- Clustering with respect to task. Clustering of users and targets are performed separately from the training. Thus the clustered result might be irrelevant to the task. We plan to build a cluster that cooperated with our classifier, for example using a variation of LDA model [BNJ03, TM08, LH09, LHZ10, BGR10, JO11].
- In the future, more complex user and target features should be considered to better describe them. For example, modelling the user's hometown as a hierarchical structure where higher level denotes wider range of area.
- The decoding speed could be a barrier for our methods, thus we will develop more scalable decoding than easiest-first decoding and more accurate decoding than two-stage decoding. For example, dual decom-

position [KRC⁺10] is a good option while it split a big dataset into smaller portions and decoding in each of them.

Bibliography

- [AB06] Alina Andreevskaia and Sabine Bergler. Mining wordnet for a fuzzy sentiment: Sentiment tag extraction from wordnet glosses. In *EACL*, 2006.
- [BCM⁺12] Farah Benamara, Baptiste Chardon, Yannick Mathieu, Vladimir Popescu, and Nicholas Asher. How do negation and modality impact on opinions? In *Proceedings of the Workshop on Extra-Propositional Aspects of Meaning in Computational Linguistics*, pages 10–18, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.
- [BDP07] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 440–447, Prague, Czech Republic, 2007.
- [BES10] Stefano Baccianella, Andrea Esuli, and Fabrizio Sebastiani. Sentiwordnet 3.0: An enhanced lexical resource for sentiment analysis and opinion mining. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the Seventh conference on International Lan-*

- guage Resources and Evaluation (LREC'10)*, Valletta, Malta, may 2010. European Language Resources Association (ELRA).
- [BGR10] Jordan Boyd-Graber and Philip Resnik. Holistic sentiment analysis across languages: Multilingual supervised latent dirichlet allocation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 45–55, Cambridge, MA, October 2010. Association for Computational Linguistics.
- [Bis06] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2006.
- [BNJ03] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, March 2003.
- [Car05] N.R. Carlson. *Psychology: The Science of Behaviour*. Pearson Education Canada, 2005.
- [Cho57] Noam Chomsky. *Syntactic Structures*. Mouton, The Hague, 1957.
- [DCP08] Mark Dredze, Koby Crammer, and Fernando Pereira. Confidence-weighted linear classification. In *Proceedings of ICML*, pages 264–271, Helsinki, Finland, 2008.
- [DS11] Adnan Duric and Fei Song. Feature selection for sentiment analysis based on content and syntax models. In *Proceedings of the 2nd Workshop on Computational Approaches to Subjectivity and*

- Sentiment Analysis (WASSA 2.011)*, pages 96–103, Portland, Oregon, June 2011. Association for Computational Linguistics.
- [ES07] Andrea Esuli and Fabrizio Sebastiani. Pageranking wordnet synsets: An application to opinion mining. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 424–431, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
- [GvDB12] Goran Glavaš, Jan Šnajder, and Bojana Dalbelo Bašić. Experiments on hybrid corpus-based sentiment lexicon acquisition. In *Proceedings of the Workshop on Innovative Hybrid Approaches to the Processing of Textual Data*, pages 1–9, Avignon, France, April 2012. Association for Computational Linguistics.
- [HL04] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *Proceedings of KDD*, pages 168–177, Seattle, WA, USA, 2004.
- [HYW⁺14] Minlie Huang, Borui Ye, Yichen Wang, Haiqiang Chen, Junjun Cheng, and Xiaoyan Zhu. New word detection for sentiment analysis. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 531–541, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [JG10] Niklas Jakob and Iryna Gurevych. Extracting opinion targets in a single and cross-domain setting with conditional random fields. In *Proceedings of the 2010 Conference on Empirical Methods in*

Natural Language Processing, pages 1035–1045, Cambridge, MA, October 2010. Association for Computational Linguistics.

- [JM09] Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2Nd Edition)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 2009.
- [JO11] Yohan Jo and Alice H. Oh. Aspect and sentiment unification model for online review analysis. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining, WSDM '11*, pages 815–824, New York, NY, USA, 2011. ACM.
- [JYM09] Lifeng Jia, Clement Yu, and Weiyi Meng. The effect of negation on sentiment analysis and retrieval effectiveness. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, pages 1827–1830, New York, NY, USA, 2009. ACM.
- [JYZ⁺11] Long Jiang, Mo Yu, Ming Zhou, Xiaohua Liu, and Tiejun Zhao. Target-dependent twitter sentiment classification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 151–160, Portland, Oregon, USA, June 2011. Association for Computational Linguistics.
- [KF09] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.
- [KI06] Alistair Kennedy and Diana Inkpen. Sentiment classification of movie reviews using contextual valence shifters. *Computational Intelligence*, 22(2):110–125, 2006.

- [KM06] Vijay Krishnan and Christopher D. Manning. An effective two-stage model for exploiting non-local dependencies in named entity recognition. In *Proceedings of COLING-ACL*, pages 1121–1128, Sydney, NSW, Australia, 2006.
- [KN06] Hiroshi Kanayama and Tetsuya Nasukawa. Fully automatic lexicon expansion for domain-oriented sentiment analysis. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, pages 355–363. Association for Computational Linguistics, 2006.
- [KRC⁺10] Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. Dual decomposition for parsing with non-projective head automata. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1288–1298, Cambridge, MA, October 2010. Association for Computational Linguistics.
- [Kup92] Julian Kupiec. Robust part-of-speech tagging using a hidden markov model. *Computer Speech & Language*, 6(3):225 – 242, 1992.
- [LH09] Chenghua Lin and Yulan He. Joint sentiment/topic model for sentiment analysis. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management, CIKM '09*, pages 375–384, New York, NY, USA, 2009. ACM.
- [LHH⁺10] Fangtao Li, Chao Han, Minlie Huang, Xiaoyan Zhu, Ying-Ju Xia, Shu Zhang, and Hao Yu. Structure-aware review mining and summarization. In *Proceedings of the 23rd International*

- Conference on Computational Linguistics*, pages 653–661, Beijing, China, August 2010. Coling 2010 Organizing Committee.
- [LHZ10] Fangtao Li, Minlie Huang, and Xiaoyan Zhu. Sentiment analysis with global topics and local dependency. In *AAAI*, 2010.
- [Liu10] Bing Liu. Sentiment analysis and subjectivity. In *Handbook of Natural Language Processing, Second Edition*. Taylor and Francis Group, Boca, 2010.
- [Liu12] Bing Liu. *Sentiment Analysis and Opinion Mining*. Synthesis digital library of engineering and computer science. Morgan & Claypool, 2012.
- [LLC⁺10] Shoushan Li, Sophia Y. M. Lee, Ying Chen, Chu-Ren Huang, and Guodong Zhou. Sentiment classification and polarity shifting. In *Proceedings of COLING*, pages 635–643, Beijing, China, 2010.
- [LRO12] Emanuele Lapponi, Jonathon Read, and Lilja Ovrelid. Representing and resolving negation for sentiment analysis. In *Data Mining Workshops (ICDMW), 2012 IEEE 12th International Conference on*, pages 687–692, Dec 2012.
- [LS09] Jingjing Liu and Stephanie Seneff. Review sentiment scoring via a parse-and-paraphrase paradigm. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 161–169, Singapore, August 2009. Association for Computational Linguistics.
- [LTS13] Angeliki Lazaridou, Ivan Titov, and Caroline Sporleder. A bayesian model for joint unsupervised induction of sentiment,

- aspect and discourse representations. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1630–1639, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
- [LXZ14] Kang Liu, Liheng Xu, and Jun Zhao. Extracting opinion targets and opinion words from online reviews with graph co-ranking. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 314–324, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [Mar83] Ian Marshall. Choice of grammatical word-class without global syntactic analysis: Tagging words in the lob corpus. *Computers and the Humanities*, 17(3):139–150, 1983.
- [MDP⁺11] Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150, Portland, Oregon, USA, 2011.
- [MF09] Justin Martineau and Tim Finin. Delta tfidf: An improved feature space for sentiment analysis. In *ICWSM*, 2009.
- [MHN⁺07] Ryan McDonald, Kerry Hannan, Tyler Neylon, Mike Wells, and Jeff Reynar. Structured models for fine-to-coarse sentiment analysis. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 432–439, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

- [ML07] Yi Mao and Guy Lebanon. Isotonic conditional random fields and local sentiment flow. In *Advances in Neural Information Processing Systems*, Vancouver, B.C., Canada, Dec 2007.
- [MP07] Karo Moilanen and Stephen Pulman. Sentiment composition. In *Proceedings of Recent Advances in Natural Language Processing*, pages 378–382, September 2007.
- [MsJ99] Kevin P. Murphy, Yair @ss, and Michael I. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence, UAI'99*, pages 467–475, San Francisco, CA, USA, 1999. Morgan Kaufmann Publishers Inc.
- [MW10] Tengfei Ma and Xiaojun Wan. Opinion target extraction in chinese news comments. In *Coling 2010: Posters*, pages 782–790, Beijing, China, August 2010. Coling 2010 Organizing Committee.
- [NIK10] Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. Dependency tree-based sentiment classification using crfs with hidden variables. In *Proceedings of NAACL-HLT*, pages 786–794, Los Angeles, CA, USA, 2010.
- [PL04] Bo Pang and Lillian Lee. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278, Barcelona, Spain, 2004.

- [PL08] Bo Pang and Lillian Lee. Opinion mining and sentiment analysis. *Foundation and Trends in Information Retrieval*, 2(1-2):1–135, 2008.
- [PLV02] Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, Pennsylvania, PA, USA, 2002.
- [PT10] Georgios Paltoglou and Mike Thelwall. A study of information retrieval weighting schemes for sentiment analysis. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1386–1395, Uppsala, Sweden, July 2010. Association for Computational Linguistics.
- [PZ04] Livia Polanyi and Annie Zaenen. Contextual valence shifters. In *In Exploring Attitude and Affect in Text: Theories and Applications (AAAI Spring Symposium Series)*, 2004.
- [QLBC11] Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27, 2011.
- [SM06] Charles Sutton and Andrew McCallum. An introduction to conditional random fields for relational learning. *Introduction to statistical relational learning*, pages 93–128, 2006.
- [SM12] Charles Sutton and Andrew McCallum. An introduction to conditional random fields. *Foundations and Trends in Machine Learning*, 4(4):267–373, 2012.

- [SME06] Kim Soo-Min and Hovy Eduard. Extracting opinions, opinion holders, and topics expressed in online news media text. In *Proceedings of ACL Workshop on Sentiment and Subjectivity in Text*, Sydney, Australia, 2006. Association for Computational Linguistics.
- [SPH⁺11] Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161, Edinburgh, Scotland, UK., July 2011.
- [SPW⁺13] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP*, pages 1631–1642, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.
- [SSUB11] Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. Twitter polarity classification with label propagation over lexical links and the follower graph. In *Proceedings of EMNLP, workshop on Unsupervised Learning in NLP*, pages 53–63, Edinburgh, UK, 2011.
- [TBT⁺11] Maite Taboada, Julian Brooke, Milan Tofiloski, Kimberly Voll, and Manfred Stede. Lexicon-based methods for sentiment analysis. *Comput. Linguist.*, 37(2):267–307, June 2011.

- [TE13] Rakshit Trivedi and Jacob Eisenstein. Discourse connectors for latent subjectivity in sentiment analysis. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 808–813, Atlanta, Georgia, June 2013. Association for Computational Linguistics.
- [TKMS03] Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL*, pages 173–180, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics.
- [TLT⁺11] Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. User-level sentiment analysis incorporating social networks. In *Proceedings of KDD*, pages 1397–1405, San Diego, California, USA, 2011.
- [TM08] Ivan Titov and Ryan McDonald. A joint model of text and aspect ratings for sentiment summarization. In *Proceedings of ACL-08: HLT*, pages 308–316, Columbus, Ohio, June 2008. Association for Computational Linguistics.
- [TM11] Oscar Täckström and Ryan McDonald. 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, Heidelberg, 2011.

- [TT05] Yoshimasa Tsuruoka and Jun'ichi Tsujii. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *In Proceedings of HLT-EMNLP*, pages 467–474, Vancouver, B.C., Canada, 2005.
- [Tur02] Peter D. Turney. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*, pages 417–424, Pennsylvania, PA, USA, 2002.
- [WBR⁺10] Michael Wiegand, Alexandra Balahur, Benjamin Roth, Dietrich Klakow, and Andrés Montoyo. A survey on the role of negation in sentiment analysis. In *Proceedings of the Workshop on Negation and Speculation in Natural Language Processing*, pages 60–68, Uppsala, Sweden, July 2010. University of Antwerp.
- [WHS⁺05] Theresa Wilson, Paul Hoffmann, Swapna Somasundaran, Jason Kessler, Janyce Wiebe, Yejin Choi, Claire Cardie, Ellen Riloff, and Siddharth Patwardhan. Opinionfinder: A system for subjectivity analysis. In *Proceedings of HLT/EMNLP 2005 Interactive Demonstrations*, pages 34–35, Vancouver, British Columbia, Canada, October 2005. Association for Computational Linguistics.
- [WWH09] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity: An exploration of features for phrase-level sentiment analysis. *Computational Linguistics*, pages 399–433, 2009.

- [WZHW09] Yuanbin Wu, Qi Zhang, Xuangjing Huang, and Lide Wu. Phrase dependency parsing for opinion mining. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1533–1541, Singapore, August 2009. Association for Computational Linguistics.
- [YC13] Bishan Yang and Claire Cardie. Joint inference for fine-grained opinion extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1640–1649, Sofia, Bulgaria, August 2013. Association for Computational Linguistics.
- [YC14] Bishan Yang and Claire Cardie. Context-aware learning for sentence-level sentiment analysis with posterior regularization. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 325–335, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [ZGMK14] Xiaodan Zhu, Hongyu Guo, Saif Mohammad, and Svetlana Kiritchenko. An empirical study on the effect of negation words on sentiment. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 304–313, Baltimore, Maryland, June 2014. Association for Computational Linguistics.
- [ZJZ06] Li Zhuang, Feng Jing, and Xiao-Yan Zhu. Movie review mining and summarization. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management, CIKM '06*, pages 43–50, New York, NY, USA, 2006. ACM.

- [ZLG⁺11] Lanjun Zhou, Binyang Li, Wei Gao, Zhongyu Wei, and Kam-Fai Wong. Unsupervised discovery of discourse relations for eliminating intra-sentence polarity ambiguities. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 162–171, Edinburgh, Scotland, UK., July 2011. Association for Computational Linguistics.
- [ZS14] Zhe Zhang and Munindar P. Singh. Renew: A semi-supervised framework for generating domain-specific lexicons and sentiment analysis. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 542–551, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

List of Publications

Journal

- Wenliang Gao, Nobuhiro Kaji, Naoki Yoshinaga and Masaru Kitsuregawa. Collective Sentiment Classification Based on User Leniency and Product Popularity. *Journal of Natural Language Processing (Volume 21 Number 3)*, pages 541-562. June 2014. (peer reviewed)

International Conference

- Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji and Masaru Kitsuregawa. Collective Sentiment Classification Based on User Leniency and Product Popularity. In *Proceedings of 27th Pacific Asia Conference on Language, Information, and Computation (PACLIC)*, pages 357-365, Taipei, Taiwan. 2013. (peer reviewed)
- Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji and Masaru Kitsuregawa. Modeling User Leniency and Product Popularity for Sentiment Classification. In *Proceedings of International Joint Conference on Natural Language Processing (IJCNLP)*, pages 1107-1111, Nagoya, Japan. 2013. (peer reviewed)

Domestic Conference

- Wenliang Gao, Nobuhiro Kaji, Naoki Yoshinaga and Masaru Kitsuregawa. Sentiment Classification by Capturing User Preferences across Targets. In *Proceedings of The 75th National Convention of IPSJ*, pages 2-89-2-90, Sendai, Japan. 2013.