Collective Sentiment Classification based on User Leniency and Product Popularity

Wenliang Gao[*], **Naoki Yoshinaga**[†], **Nobuhiro Kaji**[†] **and Masaru Kitsuregawa**[†‡]
[*]Graduate School of Information Science and Technology, The University of Tokyo
[†]Institute of Industrial Science, The University of Tokyo
[‡]National Institute of Informatics
{wl-gao, ynaga, kaji, kitsure}@tkl.iis.u-tokyo.ac.jp

Abstract

We propose a method of collective sentiment classification that assumes dependencies among labels of an input set of reviews. The key observation behind our method is that the distribution of polarity labels over reviews written by each user or written on each product is often skewed in the real world; intolerant users tend to report complaints while popular products are likely to receive praise. We encode these characteristics of users and products (referred to as *user leniency* and *product popularity*) by introducing global features in supervised learning. To resolve dependencies among labels of a given set of reviews, we explore two approximated decoding algorithms, "easiest-first decoding" and "two-stage decoding". Experimental results on two real-world datasets with product and user/product information confirmed that our method contributed greatly to the classification accuracy.

1 Introduction

In document-level sentiment classification, early studies have exploited language-based clues (e.g., n-grams) extracted from the textual content (Turney, 2002; Pang et al., 2002), followed by recent studies which adapt the classifier to the reviews written by a specific user or written on a specific product (Tan et al., 2011; Seroussi et al., 2010; Speriosu et al., 2011; Li et al., 2011). Although the user- and product-aware methods exhibited better performance over the methods based on purely textual clues, most of them use only the user information (Tan et al., 2011; Seroussi et al., 2010; Speriosu et al., 2011), or they assume that the user and the product of a test review is known in advance (Li et al., 2011). These assumptions heavily limit their applicability in a real-world scenario where new users and new products are ceaselessly emerging.

This paper proposes a method of collective sentiment classification that is aware of the user and the product of the target review, which benefits from the skewed distributions of polarity labels: intolerant users tend to report complaints while popular products are likely to receive praise. We introduce global features to encode these characteristics of a user and a product (referred to as user leniency and product popularity), and then compute the values of global features along with testing. Our method is therefore applicable to reviews written by users and on products that are not observed in the training data.

Because global features depend on labels of test reviews while the labels reversely depend on the global features, we need to globally optimize a label configuration for a given set of reviews. In this study, we resort to approximate algorithms, easiest-first (Tsuruoka and Tsujii, 2005) and two-stage strategies (Krishnan and Manning, 2006), in decoding labels, and empirically compare their speed and accuracy.

We evaluated our method on two real-world datasets with product (Maas et al., 2011) and user/product information (Blitzer et al., 2007). Experimental results demonstrated that the collective sentiment classification significantly improved the classification accuracy against the state-of-the-art methods, regardless of the choice of decoding strategy.

The remainder of this paper is organized as follows. Section 2 discusses related work that exploits user and product information in a sentiment classification task. Then, Section 3 proposes a method that collectively classifies polarity of given set of reviews. Section 4 reports experimental results. Finally, Section 5 concludes this study and addresses future work.

2 Related Work

Early studies on sentiment analysis considers only textual content for classifying the sentiment of a given review (Pang and Lee, 2008). Pang *et al.* (2002) developed a supervised sentiment classifier which only takes n-gram features. Nakagawa *et al.* (2010) and Socher *et al.* (2011) considered structural interaction among words to capture complex intra-sentential phenomena such as polarity shifting (Li et al., 2010).

On the other hand, recent studies started exploring the effectiveness of user and/or product information. Tan *et al.,* (2011) and Speriosu *et al.,* (2011) exploited user network behind a social media (Twitter in their case) and assumed that friends give similar ratings towards similar products. Seroussi *et al.* (2010) proposed a framework that computes users' similarity on the basis of text and their rating histories. Then, they classify a given review by referring to ratings given for the same product by other users who are similar to the user in question. However, such user networks are not always available in the real world.

Li *et al.* (2011) incorporate user- or product-dependent n-gram features into a classifier. They argue that users use a personalized language to express their sentiment, while the sentiment toward a product is described by product-specific language. This approach, however, requires the training data to contain reviews written by test users and written for test products. This is infeasible since labeling reviews requires too much manual work.

3 Method

This section describes our method of collective sentiment classification that uses user leniency and product popularity.

3.1 Overview

Our task is, given a set of N reviews $\mathcal{R}$, to predict labels $\mathcal{Y}$, where $y_r \in \{+1, -1\}$[1] for each given review $r \in \mathcal{R}$. The label of each review is predicted based on the following scoring function:

$$s_r = score(\boldsymbol{x}_r) = \boldsymbol{w}^T \boldsymbol{x}_r, \qquad (1)$$

[1]The labels, +1 and -1, represent positive and negative polarity, respectively.

where $\boldsymbol{x}_r$ is feature vector representation of the review r and $\boldsymbol{w}$ is the weight vector. With this scoring function, the label is predicted as follows:

$$y_r = \mathrm{sgn}(s_r) = \begin{cases} +1 & if\ s_r > 0, \\ -1 & otherwise. \end{cases}$$

Our interest is to exploit user leniency and product popularity for improving sentiment classification. We realize this by encoding such biases as two global features, as detailed in Section 3.2. Since global features make it impossible to independently predict the labels of reviews, we explored two approximate decoding strategies in Section 3.3.

Note that we assume the review is associated with the user who wrote that review, the product on which that review is written, or both. This assumption is not unrealistic nowadays. User information is available in many standard dataset (Blitzer et al., 2007; Pang and Lee, 2004). Moreover, as for product information, even if such information is not available, it is possible to extract it (Qiu et al., 2011). We should emphasize here that our method does not require user profiles, product descriptions, or any sort of extrinsic knowledge on the users and the products.

3.2 Features

Our features can be divided into local and global ones such that $\boldsymbol{x}_r = \{\boldsymbol{x}_r^l, \boldsymbol{x}_r^g\}$. While local features ($\boldsymbol{x}_r^l$) are conventional word n-grams ($n = 1$ and $n = 2$), global features ($\boldsymbol{x}_r^g$) represent the user leniency and product popularity.

Our global features are computed as:

$$\begin{aligned} \boldsymbol{x}_r^g &= \{f\_u^+(r), f\_u^-(r), f\_p^+(r), f\_p^-(r)\}, \\ & where \\ f\_u^+(r) &= \frac{|\{r_j \,|\, y_j = +1, r_j \in \mathcal{N}_u(r)\}|}{|\mathcal{N}_u(r)|}, \\ f\_u^-(r) &= \frac{|\{r_j \,|\, y_j = -1, r_j \in \mathcal{N}_u(r)\}|}{|\mathcal{N}_u(r)|}, \\ f\_p^+(r) &= \frac{|\{r_j \,|\, y_j = +1, r_j \in \mathcal{N}_p(r)\}|}{|\mathcal{N}_p(r)|}, \\ f\_p^-(r) &= \frac{|\{r_j \,|\, y_j = -1, r_j \in \mathcal{N}_p(r)\}|}{|\mathcal{N}_p(r)|}. \end{aligned}$$

$\mathcal{N}_u(r)$, the user-related neighbors, is the set of reviews, excluding r, written by the user who wrote the review r, and $\mathcal{N}_p(r)$, the product-related neighbors, is the set of reviews, excluding r, on the same product as the review r, respectively.

Algorithm 1 Easiest-first strategy

```
 1: for $r \in \mathcal{R}$ do
 2:     initialize the global features to 0
 3:     compute $score(\boldsymbol{x}_r)$
 4: while $\mathcal{R} \neq \emptyset$ do
 5:     $r_{max} = \arg\max_{r_i \in \mathcal{R}} |score(\boldsymbol{x}_{r_i})|$
 6:     $y_{r_{max}} = \mathrm{sgn}(score(\boldsymbol{x}_{r_{max}}))$
 7:     for $r_j \in (\mathcal{N}_u(r_{max}) \cup \mathcal{N}_p(r_{max})) \cap \mathcal{R}$ do
 8:         update global features
 9:         re-compute $score(\boldsymbol{x}_{r_j})$
10:     $\mathcal{R} = \mathcal{R} \backslash \{r_{max}\}$
11: return $\mathcal{Y}$
```

Algorithm 2 Two-stage strategy

```
1: for $r \in \mathcal{R}$ do
2:     $y_r = \mathrm{sgn}(score(\boldsymbol{x}_r))$
3: for $r \in \mathcal{R}$ do
4:     compute global features
5:     $y_r = \mathrm{sgn}(score(\boldsymbol{x}_r))$
6: return $\mathcal{Y}$
```

The first two features capture user leniency, i.e., how likely the user is to write positive and negative reviews, respectively. The other features capture product popularity, i.e., how likely positive and negative reviews on the product at hand are to be written.

3.3 Two Approximate Decoding Strategies

The global features make it difficult to perform decoding, i.e., labeling reviews, since each review 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: easy-first (Tsuruoka and Tsujii, 2005) and two-stage strategy (Krishnan and Manning, 2006).

Algorithm 1 depicts the easiest-first decoding algorithm. This strategy iteratively determines the label of each review one by one. In each iteration step, a review that is the easiest to label , i.e., the review with the highest score, is picked up (line 5 in Algorithm 1), and then its label is determined (line 6 in Algorithm 1). This process is repeated until all the reviews are labeled. The global features are computed by using the labels of reviews that are already assigned with labels. That is, at the beginning of decoding, no global features are fired; more global features are fired as the labeling process proceeds. The score of the review is computed in a different way depending on how global features are fired, as analogous to (Tsuruoka and Tsujii, 2005). Specifically, we prepare four classifiers, and those classifiers are used when (1) no global features are fired, (2) only user leniency features are fired, (3) only product popularity features are fired, and (4) both global features are fired, respectively.

Next, we introduce a two-stage strategy (Krishnan and Manning, 2006), which has better scalability than easy-first strategy. It is depicted in Algorithm 2. This strategy performs decoding twice. In the first stage (line 1 to line 2 in Algorithm 2), we ignore all the global features, and use only local features to classify all the reviews. In the second stage (line 3 to line 5 in Algorithm 2), labels predicted in the first stage are used to compute global features and the labels are re-assigned by using both global features and local features. In our case, two-stage at first only uses word n-gram features to estimate the labels of reviews. Thereafter, those labels are used to compute global features in the second stage.

3.4 Time Complexity

This subsection analyzes the time complexity of the two decoding strategy with respect to the number of reviews, N.

In easiest-first strategy, two processes consume most of the computing time. One is choosing the easiest review label (line 5 in Algorithm 1). The $arg\,max$ operation takes $O(\log \mathcal{N})$ time in each iteration by using a heap structure. Thus, the total time complexity in this step is $O(\mathcal{N} \log \mathcal{N})$ for N iteration. Another bottleneck is score recomputation (line 9 in Algorithm 1). To update the score for each review $r_j \in \mathcal{N}_u(r_{max}) \cap \mathcal{N}_p(r_{max})$, we need at most $|\mathcal{N}_u(r_{max}) \cap \mathcal{N}_p(r_{max})|$ times *delete* and *insert* operations to the heap. Since we could limit the number of reviews for each user or each product, $|\mathcal{N}_u(r_{max}) \cap \mathcal{N}_p(r_{max})|$ is treated as a constant C.[2] The overall time complexity sums up to $O(\mathcal{N}(\log \mathcal{N} + C \log \mathcal{N})) = O(\mathcal{N} \log \mathcal{N})$.

In 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

[2]However, based on our experiment as shown in Figure 2, the number $|\mathcal{N}_u(r_{max}) \cap \mathcal{N}_p(r_{max})|$ is weakly related to N.

Dataset	Blitzer	Maas
No. of reviews	188,350	50,000
No. of users	123,584	n/a
No. of products	101,021	7,036
No. of reviews/user	1.5	n/a
No. of reviews/products	1.9	7.1

Table 1: Dataset statistics.

that uses only local textual features.

3.5 Training

It is straightforward to train the parameters of the scoring functions. We train a binary classifier as the score estimation function in Eq. 1, considering word n-gram features, user leniency features, and product popularity features. The values of global features are computed by using the gold labels. We assume that a value of the user leniency feature or product popularity feature for a review whose user has no other reviews or whose product has no other reviews is set to 0.

4 Experiments

In this section, we evaluate our method of collective sentiment classification on two real-world review datasets with user/product or product information (Blitzer et al., 2007; Maas et al., 2011).

We preprocessed each review in the datasets by OpenNLP[3] toolkit to detect sentence boundaries and to tokenize n-grams. Following Pang *et al.* (2002), we induce word unigrams and bigrams as local features, taking negation into account. We ignored n-grams that appeared less than six times in the training data.

We adopted a confidence-weighted linear classifier (Dredze et al., 2008) with n-gram features as our baseline. To make the comparison fair, we used the same classifier, which despite of local features also considers global features, as the local classifier in our method. We used the default hyper-parameters to this classifier. Note that the confidence-weighted algorithm performed as good as SVM (Dredze et al., 2008) so it constructs a strong baseline.

4.1 Datasets

Blitzer *et al.* (2007) and Maas *et al.* (2011) collected two datasets which contain user/product or product information respectively. Table 1 summarizes the statistics of the two datasets. We should mention that the original Blitzer dataset contains more than 780k reviews collected from Amazon.com on several domains (e.g. books, movies and games). We automatically delete replicated reviews 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) to maintain consistency with the Maas dataset.

The Maas dataset has 25,000 positive and 25,000 negative reviews on movies. We have used a URL (linked to the move title) provided with each review as the identifier of the product movie. Because user information cannot be fully recovered, we only model the product popularity on this dataset.

In the two datasets, the reviews were ordered by

Method	Blitzer	Maas
Seroussi *et al.,* (2010)	89.37	n/a
Maas *et al.,* (2011)	n/a	88.89[4]
baseline	90.13	91.41
proposed (easiest-first)		
+user	91.04$^{\gg}$	n/a
+product	90.16$^{>}$	**92.73**
+user +product	**91.11**$^{\gg}$	n/a
proposed (two-stage)[5]		
+user	90.95$^{\gg}$	n/a
+product	90.15	92.68
+user +product	91.02$^{\gg}$	n/a

Table 2: Accuracy (%) on review datasets. *+user*/*+product* means modeling the user leniency / product popularity features. Accuracy marked with "$\gg$" or "$>$" was significantly better than baseline ($p < 0.01$ or $0.01 \le p < 0.05$ assessed by McNemar's test).

[3]`http://opennlp.apache.org/`

[4]This results uses different 2-fold splitting from ours. Under their splitting, our accuracies (+user+product) are 91.02%, 92.54% and 92.28% for baseline, easiest-first and two-stage with product popularity features respectively. Both strategies easily beat Maas *et al.,* (2011)'s accuracy, 88.89%. The main difference between our baseline and their baseline is the features. They use only unigram features (baseline accuracy is 87.80%), while we use unigram and bigram (which considers negation) as features.

[5]The two-stage implementation in Gao *et al.* (2013) used a different setting. In that paper, the classifiers for the first stage and second stage are the same one considering local and global features. While in this paper, the classifier used in the first stage only considers local features and the classifier for the second stage considers both.

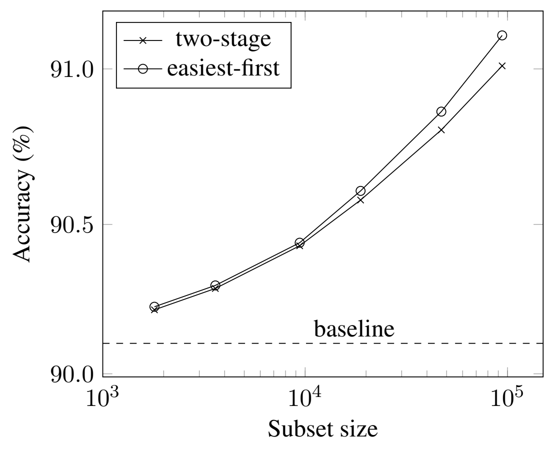

Figure 1: Average accuracy when we changed the size of subset on test reviews.

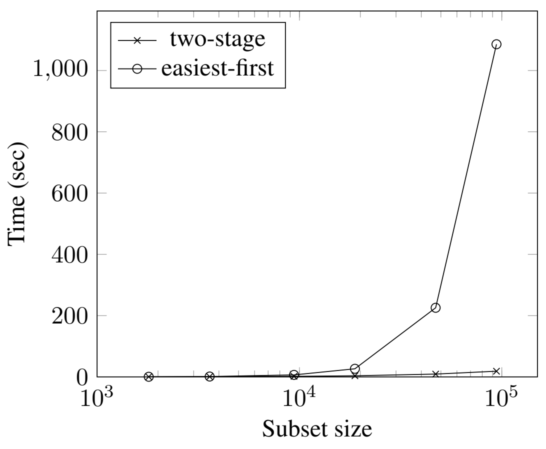

Figure 2: Average computation time when we changed the size of subset on test reviews.

user and product. In order to prevent the seemingly unfair accuracy gain under this particular splitting, we performed a 2-fold cross-validation after randomly splitting reviews, rather than using the split provided by the authors.

4.2 Results

We then compared the accuracy of our method with the two baseline methods on the two datasets: a confidence-weighted linear classifier with n-gram features and a user-aware sentiment classifier proposed by Seroussi *et al.* (2010).

In Seroussi's method, we need to fix the threshold to the number of reviews written by the same user to prepare and train a personalized classifier. After several test, the threshold is set to be 5 to gain a better performance[6]. Similarity of users is computed by word n-gram jaccard distance (called "AIT" in their paper). When the user of the test review is unseen in the training set, the default classifier trained on all the training reviews (identical to the other baseline classifier based on n-grams) is used to determine the label.

Table 2 shows the experimental results. Our method significantly improved accuracies across the two datasets against the baseline classifier. A larger improvement is acquired on the Maas dataset probably because the average number of reviews for each product is higher than that on the Blitzer dataset so we could estimate more reliable global features.

On the Blitzer dataset, the user leniency was more helpful than the product popularity. This is probably because the Blitzer dataset had been collected for users, which means to collect all the reviews written by each user. While on the Maas dataset, product information plays a important role because the reviews are collected for each product.

Among the two decoding methods, the easiest-first decoding achieved better for this test. This conforms our expectation that the easiest-first decoding is more cautious than the other. However, easiest-first decoding has it's own weakness. In what follows, we investigate the speed and accuracy trade-off of the two decoding methods.

Impact of test review size on speed and accuracy: Next, we investigate the impact of the number of test reviews on speed and accuracy in our collective sentiment classification. We use Blitzer dataset for evaluation because of its larger review size. The two types of global features are both considered.

We performed 2-fold cross-validation with the same splitting for the Blitzer dataset, while changing the size of test reviews processed at once to investigate the impact of test review size on classification accuracy. In this experiment, we split the test reviews into equal-sized smaller subsets and applied our classifier independently to each of the subsets. We average the result for all the subsets to get a stable accuracy. Figure 1 shows the experimental results. When we process a larger number of reviews at once, the accuracies of the two methods increase because of more reliable global features.

We then performed the speed test using the same setting as the previous test, but measured the average time consumed by one single subset. As

[6]Seroussi *et al.,* chose users who have more than 50 positive and 50 negative reviews. Few users or product in

	(sp)	(up)	total
No. of reviews	46,397	3,603	50,000
Ave. No. of reviews/product	4.82	1.62	4.22
baseline	91.87	85.51	91.41
proposed (easiest-first)	93.11 (+1.24)	87.73 (+2.22)	92.73 (+1.32)
proposed (two-stage)	93.09 (+1.22)	87.48 (+1.97)	92.68 (+1.21)

Table 3: Accuracy (%) on known/unknown product splits on Maas dataset. *sp* and *up* stand for *seen product* and *unseen product*. Float inside parenthesizes is the difference compared to the baseline classifier.

	(su, sp)	(uu, sp)	(su, up)	(uu, up)	total
No. of reviews	35,689	60,775	36,895	55,027	188,350
Ave. No. of reviews/user	2.04	1.04	2.14	1.04	1.40
Ave. No. of reviews/product	1.20	1.39	1.14	1.20	1.43
baseline	89.71	90.45	90.37	89.95	90.13
proposed (easiest-first)	91.42 (+1.71)	90.93 (+0.59)	92.19 (+1.82)	90.76 (+0.81)	91.11 (+0.98)
proposed (two-stage)	91.23 (+1.52)	90.88 (+0.54)	92.09 (+1.72)	90.30 (+0.35)	91.02 (+0.89)

Table 4: Accuracy (%) on known/unknown user/product splits on Blitzer dataset. *su*, *uu*, *sp* and *up* stand for *seen user*, *unseen user*, *seen product* and *unseen product* respectively. Float inside parenthesises is the difference compared to the baseline classifier.

shown in Figure 2, the speed of the easiest-first decoding significantly slows down as the number of processed reviews grows, whereas the speed of the two-stage decoding increases compute time linearly. Meanwhile, the accuracy of the two strategies are competitive as shown in Figure 1.

These results confirm the analysis in Section 3.4 that the easiest-first decoding takes most of the time in re-computing and sorting the scores. More specifically, if the user has plenty of reviews or the product has been rated by plenty of reviews, the score frequently changes in each iteration in response to the change of global features' values. Based on these observations, when the amount of test data is large, the two-stage decoding is tremendously faster with only a little loss of accuracy. When the dataset is small, to fully utilize the user leniency and product popularity properties, easiest-first decoding should be adopted.

Impact of user/product-awareness: We investigate the performance on the test reviews when we observed the user/product or not in the training data. We use the leniency and popularity global features on the Blitzer dataset, 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 known user or written on known product because similar *n*-grams would be contained in the training. On the other hand, in our model's setting, more reviews per user (or per product) should lead to more reliable leniency (or popularity) features thus better accuracy.

On the Maas dataset as shown in Table 3, the improvement on unknown product set is larger than that on known product set. We have to note here that the improvement on the unknown product set is greater while the review number for each product is smaller, which seems to violate our assumption. The reason is that baseline on the unknown product set performed poorly, which left our method larger space for improvement, even without enough global features.

On the Blitzer dataset as shown in Table 4, improvement is higher on known user sets. We find that average review number for each user is extremely low (1.04 reviews). Then lacking reliable global features may be the main reason for the poor performance on unknown user sets. We next investigate how many reviews are needed to compute reliable global features.

Accuracy Distribution: We here investigate how the reliability of the global features would influence the accuracy improvement. We exploit the accuracies with respect to how many reviews

No. of user-related neighbors ($\|\mathcal{N}_u(r)\|$)	No. of product-related neighbors ($\|\mathcal{N}_p(r)\|$) 0	1	2	3-
0	55,043 90.11 (+0.03)	34,735 90.13 (+0.26)	16,601 90.80 (+0.53)	9,630 92.48 (+0.58)
1	10,768 91.18 (+1.37)	6,530 91.24 (+2.11)	2,974 91.32 (+1.17)	1,536 92.12 (+1.04)
2	4,595 91.28 (+1.55)	2,711 91.26(+2.66)	1,292 90.56 (+1.71)	663 92.14 (+2.36)
3-7	8,120 92.48 (+2.33)	4,974 91.19 (+2.27)	2,174 92.18 (**+3.31**)	998 90.18 (+1.50)
8-	13,243 **93.73** (+2.2)	7,484 92.28 (+1.74)	3,017 91.28 (+1.52)	1,289 90.22 (+1.62)

Table 5: Accuracy (%, downer inside cell) of proposed method (two-stage) and review size (upper inside cell) on Blitzer dataset separated according to the number of reviews written by the user and the number of reviews on the product.The float inside parenthesizes is the difference from the baseline method.

No. of product-related neighbors ($\|\mathcal{N}_p(r)\|$) 0	1	2-5	6-10	11-
3,597 86.41 (+0.42)	4,646 90.94 (+1.96)	14,394 92.59 (+1.75)	10,444 93.98 (+1.31)	16,919 93.78 (+0.83)

Table 6: Accuracy (%, downer inside cell) of proposed method (two-stage) and the review size (upper inside cell) on Blitzer dataset separated according to the number of reviews on the product. The float inside parenthesizes is the difference from the baseline method.

each user or product has. More reviews means that more reliable global features will be extracted by our model.

Since user leniency is the dominant influential global feature on the Blitzer dataset, Table 5 shows the leniency features is related to the improvement. Product popularity has limited influence on this dataset because it is collected according to users. On the Maas dataset, popularity features play an important role as shown in Table 6.

We noticed that when the review number of a user or a product reaches some point ($|\mathcal{N}_u(r)| = 3-7$ in the Blitzer dataset and $|\mathcal{N}_p(r)| = 2-5$ in the Maas dataset), having more reviews does not improve the accuracy any further. However, higher $|\mathcal{N}_u(r)|$ or $|\mathcal{N}_p(r)|$ number induces lower speed of easiest-first decoding as we analyzed in Section 3.4. Then, we could collect a bounded number of reviews for each user or product to cost less time and acquire better accuracy.

Examples: Some examples are given here to explain how our model would work. As shown in Table 7, it is sometimes hard to correctly classify labels when only the text is given.

In the first two examples, weak negative textual features are found in the test instance. However, since the two users are lenient and the first product 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 latent meaning of the words (sometimes, even real person feels hard to extract sentiment from these words). As we can see in the third example in Table 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 use the user and product-based dependencies to improve sentiment classification accuracy. Nowadays, in the big data background, this method could be more useful with huge amount of unlabeled data.

5 Conclusion

We have presented collective sentiment classification which captures and utilizes user leniency and product popularity. Different from the previous studies that are aware of the user and product of

leniency	popularity	content	labels		
			golden	baseline	proposed
f_u^+: **0.92** f_u^-: 0.08	f_p^+: **0.67** f_p^-: 0.33	... The book would deserve 5 stars is the author had compared several popular jurisdictions **instead of** focusing solely on Nevada	+1	-1	+1
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 helpfulas earlier editions	+1	-1	+1
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 ...	-1	+1	-1

Table 7: Examples show the influence of leniency and popularity global features. The **bold** content is the negative evidence learned by classifier.

the review, our model does not assume the training data to contain the reviews written by the same user of test reviews or written on the same product of test reviews. To decode a labels configuration for a given set of reviews, we adopted and compared two strategies, namely "easiest-first decoding" and "two-stage decoding".

We conducted experiments on two real-world review datasets to compare our method with the existing methods. The proposed method performed more accurately than the baseline methods that uses word n-gram as features. It also outperforms another state-of-the-art method which trains personalized sentiment classifiers significantly. The more reviews per-user/product possesses, the larger improvement our model would gain. Two-stage strategy gains less accuracy than easiest-first, however, consumes only linear time in terms of the test review size (expected to be the same order of speed as the baseline classifiers). We plan to publish the code and datasets[7].

A future extension of this work is to use this on other task, such as classifying the subjectivity of a given document. We also plan to use dual decomposition as an advanced decoding strategy on our model.

References

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proceedings of ACL*, pages 440–447, Prague, Czech Republic.

Mark Dredze, Koby Crammer, and Fernando Pereira. 2008. Confidence-weighted linear classification. In *Proceedings of ICML*, pages 264–271, Helsinki, Finland.

Wenliang Gao, Naoki Yoshinaga, Nobuhiro Kaji, and Masaru Kitsuregawa. 2013. Modeling user leniency and product popularity for sentiment classification. In *Proceedings of IJCNLP*, Nagoya, Japan. to appear.

Vijay Krishnan and Christopher D. Manning. 2006. 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.

Shoushan Li, Sophia Y. M. Lee, Ying Chen, Chu-Ren Huang, and Guodong Zhou. 2010. Sentiment classification and polarity shifting. In *Proceedings of COLING*, pages 635–643, Beijing, China.

Fangtao Li, Nathan Liu, Hongwei Jin, Kai Zhao, Qiang Yang, and Xiaoyan Zhu. 2011. Incorporating reviewer and product information for review rating prediction. In *Proceedings of IJCAI*, pages 1820–1825, Barcelona, Spain.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In *Proceedings of ACL-HLT*, pages 142–150, Portland, Oregon, USA.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Proceedings of NAACL-HLT*, pages 786–794, Los Angeles, CA, USA.

Bo Pang and Lillian Lee. 2004. A sentimental education: sentiment analysis using subjectivity summarization based on minimum cuts. In *Proceedings of ACL*, pages 271–278, Barcelona, Spain.

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

[7]http://www.tkl.iis.u-tokyo.ac.jp/~wl-gao/

Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? sentiment classification using machine learning techniques. In *Proceedings of EMNLP*, pages 79–86, Pennsylvania, PA, USA.

Guang Qiu, Bing Liu, Jiajun Bu, and Chun Chen. 2011. Opinion word expansion and target extraction through double propagation. *Computational Linguistics*, 37(1):9–27.

Yanir Seroussi, Ingrid Zukerman, and Fabian Bohnert. 2010. Collaborative inference of sentiments from texts. In *Proceedings of UMAP*, pages 195–206, Big Island, HI, USA.

Richard Socher, Jeffrey Pennington, Eric H. Huang, Andrew Y. Ng, and Christopher D. Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of EMNLP*, pages 151–161, Edinburgh, Scotland, UK., July.

Michael Speriosu, Nikita Sudan, Sid Upadhyay, and Jason Baldridge. 2011. 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.

Chenhao Tan, Lillian Lee, Jie Tang, Long Jiang, Ming Zhou, and Ping Li. 2011. User-level sentiment analysis incorporating social networks. In *Proceedings of KDD*, pages 1397–1405, San Diego, California, USA.

Yoshimasa Tsuruoka and Jun'ichi Tsujii. 2005. Bidirectional inference with the easiest-first strategy for tagging sequence data. In *In Proceedings of HLT-EMNLP*, pages 467–474, Vancouver, B.C., Canada.

Peter D. Turney. 2002. Thumbs up or thumbs down?: semantic orientation applied to unsupervised classification of reviews. In *Proceedings of ACL*, pages 417–424, Pennsylvania, PA, USA.