Collective Sentiment Classification Based on User Leniency and Product Popularity

Wenliang Gao[†], Nobuhiro Kaji[††,†††], Naoki Yoshinaga[††,†††] and Masaru Kitsuregawa[††,††††]

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 real-world datasets with user and/or product information confirm that our method contributed greatly to classification accuracy.

Key Words: *sentiment classification, user leniency, product popularity, easiest-first decoding, two-stage decoding*

1 Introduction

In document-level sentiment classification, early studies exploited language-based clues (e.g., n-grams) extracted from textual content (Turney 2002; Pang, Lee, and Vaithyanathan 2002), followed by more recent studies that adapt the classifier to reviews written by a specific user or written on a specific product (Tan, Lee, Tang, Jiang, Zhou, and Li 2011; Seroussi, Zukerman, and Bohnert 2010; Speriosu, Sudan, Upadhyay, and Baldridge 2011; Li, Liu, Jin, Zhao, Yang, and Zhu 2011). Although user- and product-aware methods exhibited better performance over those 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 product of a test review are seen in the training data (Li et al. 2011). These assumptions heavily limit their applicability in a real-world scenario where new users and new products are constantly emerging.

This paper proposes a method of collective sentiment classification that is aware of the user

† Graduate School of Information Science and Technology, The University of Tokyo
†† Institute of Industrial Science, The University of Tokyo
††† National Institute of Information and Communications Technology (NICT)
†††† National Institute of Informatics
* This work was conducted while the second and third authors were at the Institute of Industrial Science, The University of Tokyo.

and product of the target review. Our method benefits from the biased distributions of polarity labels in the real world; intolerant users tend to report complaints while popular products are likely to receive praise. We introduce global features to encode the bias of a user and of a product (referred to as user leniency and product popularity), and then compute the global features along with testing. In this way, the global features are collectively computed with respect to the labels of other test reviews. Our method is therefore applicable to reviews written by emerging users and on emerging products that are not observed in the training data.

The major difficulty in realizing our collective sentiment classifier is in decoding. Because global features depend on test review labels and labels conversely depend on the global features, we need to optimize a global label configuration for the test reviews. In this study, we tackle this problem by resorting to two approximate decoding algorithms, easiest-first (Tsuruoka and Tsujii 2005) and two-stage strategies (Krishnan and Manning 2006). We also empirically compare the speed and accuracy of these two strategies.

We evaluate our methods on three datasets with user and/or product information (Pang and Lee 2004; Blitzer, Dredze, and Pereira 2007; Maas, Daly, Pham, Huang, Ng, and Potts 2011). Experimental results demonstrate that when user- or product-bias exists, our collective method can improve classification accuracy against state-of-the-art methods.

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. Section 3 proposes a method that collectively classifies a given set of reviews. Section 4 reports experimental results. Finally, Section 5 concludes this study and addresses future work.

2 Related Work

Document-level sentiment classification focuses on labeling a given review (Turney 2002). Normally, the content of the review, the user who wrote the review, and the product on which the review is written are considered contributive to this research (Pang and Lee 2008). In what follows, we briefly glance at traditional approaches based on purely textual content, and introduce user- or product-aware approaches in depth.

2.1 Text-based Methods

Early studies consider only textual content for classifying the sentiment. Pang et al. (2002) developed a supervised sentiment classifier that takes only word n-grams as features. Hu and Liu (2004) analyzed the sentiment of products' aspects. Blitzer et al. (2007) built a classifier

that learns weights of textual features depending on the product domain. Nakagawa, Inui, and Kurohashi (2010) and Socher, Pennington, Huang, Ng, and Manning (2011) considered structural interaction between words to capture complex intra-sentential phenomena such as polarity shifting (Li, Lee, Chen, Huang, and Zhou 2010). Qiu, Liu, Bu, and Chen (2011) used a bootstrap method to expand sentiment lexicon, which greatly helps to identify the sentiment.

2.2 User- or Product-Aware Methods

Recently, user-generated content has been the focus of considerable attention stimulating 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 the same products. However, such user networks are not always available in the real world.

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 were 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 employ user-specific language to express their sentiment, while the sentiment toward a product is described in 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 their characteristics from the test reviews. As we later confirm in experiments, our method improves classification accuracy even when only a few reviews are available for the users or products in question.

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 estimate labels $\mathcal{Y}$, where $y_r \in \{+1, -1\}$ for each given review $r \in \mathcal{R}$, $+1$ and -1 represent positive and negative polarity, respectively. Each

review label is estimated based on the following scoring function,

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

where $\boldsymbol{x}_r$ is feature vector representation of the review r, and $\boldsymbol{w}$ is a weight vector that we learn from labeled data. With this scoring function, the label is estimated as follows:

$$y_r = \mathrm{sgn}(score(\boldsymbol{x}_r)) = \begin{cases} +1 & \text{if } score(\boldsymbol{x}_r) > 0, \\ -1 & \text{otherwise.} \end{cases}$$

As discussed in the introduction, our aim is to exploit user leniency and product popularity to improve sentiment classification. We therefore encourage reviews written by the same user or on the same product to receive the same polarity when their polarity labels are found to be biased. We realize this by encoding such biases as two global features in addition to local textual features, as detailed in Section 3.2. Since global features make it impossible to estimate review labels independently, we explore using two approximate decoding strategies in Section 3.3.

Note that here we assume each review to be associated with either the user who wrote that review, the product on which the review was written, or both. This assumption is not unrealistic, since the user or product can be identified in many review websites. We should emphasize that our method does not require user profiles, product descriptions, or any type of extrinsic knowledge of the users or products, and therefore it can handle reviews written by emerging users or on emerging products.

3.2 Features

Our features can be divided into local and global, such that $\boldsymbol{x}_r = (\boldsymbol{x}_r^l, \boldsymbol{x}_r^g)$. The local features ($\boldsymbol{x}_r^l$) are conventional word n-grams ($n = 1$ and $n = 2$) with binary values that indicate the existence of the n-grams. The global features ($\boldsymbol{x}_r^g$) are the user leniency and product popularity that are represented as real values.

Our global features are decomposed as:

$$\boldsymbol{x}_r^g = (f\_u^+(r), f\_u^-(r), f\_p^+(r), f\_p^-(r)),$$

where

$$f\_u^+(r) = \frac{|\{r_j \,|\, y_j = +1, r_j \in \mathcal{S}_u(r)\}|}{|\mathcal{S}_u(r)|}, \quad f\_u^-(r) = \frac{|\{r_j \,|\, y_j = -1, r_j \in \mathcal{S}_u(r)\}|}{|\mathcal{S}_u(r)|},$$
$$f\_p^+(r) = \frac{|\{r_j \,|\, y_j = +1, r_j \in \mathcal{S}_p(r)\}|}{|\mathcal{S}_p(r)|}, \quad f\_p^-(r) = \frac{|\{r_j \,|\, y_j = -1, r_j \in \mathcal{S}_p(r)\}|}{|\mathcal{S}_p(r)|}.$$

Here, $\mathcal{S}_u(r)$ is the user-related neighbor set of r, which contains the reviews written by the same user u as r, while $\mathcal{S}_p(r)$ is the product-related neighbor set of r, which contains reviews written on the same product p as r. If $\mathcal{S}_u(r)$ (or $\mathcal{S}_p(r)$) is empty, we set $f\_u^+(r)$ and $f\_u^-(r)$ (or $f\_p^+(r)$ and $f\_p^-(r)$) to be 0.

We use $f\_u^+(r)$ and $f\_u^-(r)$ to capture user leniency, i.e., how likely the user is to write positive and negative reviews, respectively, while we use $f\_p^+(r)$ and $f\_p^-(r)$ to capture product popularity, i.e., how likely positive and negative reviews are written on the product, respectively.[1]

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. Instead, we explore and empirically compare two approximate algorithms, easiest-first (Tsuruoka and Tsujii 2005) and two-stage decoding strategy (Krishnan and Manning 2006). The easiest-first decoding is slower but expected to be more accurate than the two-stage decoding.

Algorithm 1 depicts the easiest-first decoding algorithm. This strategy iteratively determines review labels one by one. In each iteration, the review that is easiest to label, i.e., review r_{max} with the highest absolute score $score(\boldsymbol{x}_r)$, is chosen (line 5 in Algorithm 1), and then labeled (line 6 in Algorithm 1). This process is repeated until all the reviews are labeled. The global features are incrementally updated using the review labels that are already assigned. That is, at the beginning of decoding, all global features are set to 0; when the labeling process proceeds, the global features become more accurate as more labels are used to compute them.

Algorithm 2 depicts a two-stage decoding algorithm (Krishnan and Manning 2006). This strategy performs decoding twice. In the first stage (lines 1–3 in Algorithm 2), we use only local features to classify the reviews. In the second stage (lines 4–7 in Algorithm 2), those labels are used to compute global features, and the labels are reassigned using the additionally computed global features. In our case, the 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.

[1] Considering f_u^+ and f_u^- (f_p^+ and f_p^-) always add up to 1, we could also use only one feature for each leniency and popularity (e.g. f_u^+ and f_p^+). We ran some experiments and found that the two-feature designation outperformed the one-feature designation (f_u^+, f_p^+) on two out of three datasets (Maas and Blitzer). Therefore, we choose to represent user leniency and product popularity using positive and negative ratio of labels.

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: $r_{max} = \arg\max_{r \in \mathcal{R}} |score(\boldsymbol{x}_r)|$
6: $y_{r_{max}} = \mathrm{sgn}(score(\boldsymbol{x}_{r_{max}}))$
7: **for** $r_j \in (\mathcal{S}_u(r_{max}) \cup \mathcal{S}_p(r_{max})) \cap \mathcal{R}$ **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: $\boldsymbol{x}_r = \boldsymbol{x}_r^l$
3: $y_r = \mathrm{sgn}(score(\boldsymbol{x}_r))$
4: **for** $r \in \mathcal{R}$ **do**
5: compute global features $\boldsymbol{x}_r^g$
6: $\boldsymbol{x}_r = (\boldsymbol{x}_r^l, \boldsymbol{x}_r^g)$
7: $y_r = \mathrm{sgn}(score(\boldsymbol{x}_r))$
8: **return** $\mathcal{Y}$

The major difference in the two algorithms is in the way they compute global features. The easiest-first strategy uses labels estimated by local features and previously computed global features, while the two-stage strategy uses labels estimated only by local features. We expect the easiest-first decoding will exhibit better classification accuracy over the two-stage strategy, which we will confirm later in experiments.

Time Complexity

We here analyze the time complexity of the two decoding strategies with respect to the number of test reviews, N.

In the easiest-first strategy, two processes consume most of the computation time, one of which is choosing the easiest review to label (line 5 in Algorithm 1). The $arg\,max$ operation spends $O(\log N)$ time in each iteration, using a heap structure to maintain the scores. Thus, the time complexity of this step is $O(N \log N)$ for N iterations. Another bottleneck is score recomputation (line 9 in Algorithm 1). To update the score for each review $r \in \mathcal{S}_u(r_{max}) \cap \mathcal{S}_p(r_{max})$, we need $|\mathcal{S}_u(r_{max}) \cap \mathcal{S}_p(r_{max})|$ times *delete* and *insert* operations to the heap. If we can assume the maximal number of reviews for each user or each product, $|\mathcal{S}_u(r_{max}) \cap \mathcal{S}_p(r_{max})|$ is upper-bounded by a constant C.[2] The overall time complexity adds 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 baseline method that uses only local textual features.

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

3.4 Training

It is straightforward to train the parameters of the scoring function for the two decoding algorithms. We train a binary classifier as the score estimation function in Eq. 1, considering word n-gram, user leniency and product popularity features. The values of global features are computed using the gold labels of training data. This classifier is used for the easiest-first decoding and second stage of the two-stage decoding. A classifier used in the first stage of the two-stage decoding is trained only with word n-gram features.

4 Experiments

In this section, we evaluate our method of collective sentiment classification on three real-world review datasets with user and/or product information (Pang and Lee 2004; Blitzer et al. 2007; Maas et al. 2011).

4.1 Setting

We preprocessed each review in the datasets using OpenNLP[3] toolkit to detect sentence boundaries and to tokenize sentences. Following Pang et al. (2002), we induced word unigrams and bigrams as local features while taking negation into account. We ignored the n-grams that appeared less than a predefined number of 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 (Dredze, Crammer, and Pereira 2008) in our methods.[4] We should emphasize here that the confidence-weighted algorithm is reported to perform as well as Support Vector Machine in a document-level sentiment classification (Dredze et al. 2008), and it thereby constructs a strong baseline.

For each confidence-weighted classifier, we tune the two hyper-parameters (confidence parameter ϕ and the number of iterations for training) on the training data. Confidence-weighted learning adjusts a multivariate Gaussian distribution over the weight parameters where ϕ controls the update rate of the variance and mean. Given a larger ϕ, the variance decreases faster and the mean is updated more gradually. The number of iterations controls how many times each training instance is used to update the parameters. We divided the training data into two equal-sized parts. In tuning, one part is used as training data and the other as development data. The parameter ϕ is chosen between {1, 2, 5, 10, 20, 50} and the number of iterations is chosen

[3] http://opennlp.apache.org/
[4] The code was kindly provided by the author of this paper.

between {1, 2, 5, 10, 15, 20, 25}. After tuning, we used the tuned hyper-parameters to train a classifier with the entire training data.

4.2 Datasets

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. Table 1 summarizes the statistics of these datasets.

Pang: This dataset is a small subset of reviews manually chosen from a movie review archive[5] collected from a discussion newsgroup on art movies. 2,000 reviews are randomly chosen from a large archive that contains over 30,000 reviews.

Blitzer: This dataset is collected from a shopping website[6] on various domains of products. We used part of its total 780 k reviews, to be consistent with the other two datasets. We automatically deleted replicated reviews written by the same author on the same product (resulting in 740 k raw reviews). Then, the reviews are balanced for positive and negative labels (over 90 k reviews for each, by randomly sampling the equal number of positive and negative reviews).

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

We automatically recovered the user and product information (implicitly) included in the datasets. The Pang and Blitzer datasets are accompanied by 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 the product, in this case, a movie. Because user information cannot be fully recovered in the Maas dataset, we only

Table 1 Dataset statistics

Dataset	Pang	Blitzer	Maas
No. of reviews	2,000	188,350	50,000
No. of users	309	123,584	n/a
No. of products	1,107	101,021	7,036
No. of reviews/user	6.5	1.5	n/a
No. of reviews/products	1.8	1.9	7.1

[5] http://reviews.imdb.com/Reviews
[6] http://www.amazon.com

consider the product popularity in this dataset.

When we split the datasets for cross-validation, we maintained the order of the Pang dataset and shuffled the Blitzer and Maas 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 (as we will later confirm in Section 4.3). In the Blitzer and Maas datasets, the reviews were originally ordered by the user and product, respectively. Therefore, if we naively use the original order given by the datasets without shuffling when splitting them, the average number of reviews for each user or product becomes unnaturally high, which generates advantages to our method. In order to prevent the seemingly unfair accuracy gain in this particular splitting, we shuffled the reviews before any experiment rather than using the split provided by the authors. We performed a two-fold cross-validation on all three datasets.

4.3 Results

We compared the accuracy of our method with two other methods: a baseline method using a 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 two-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 his/her training reviews if he/she has more (positive and negative) reviews than a predefined threshold. For any pair of 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. If we set the aforementioned threshold too high, many word n-gram features will be lost because many reviews will be ignored. We tuned the threshold (from 1 to 5) using the identical method we used to tune the hyper-parameters for the confidence-weighted classifier.[7] In our datasets, many test users had a similarity of 0 to the users in the training data because the number of reviews written by each user is much smaller than that in the Seroussi's dataset. For labeling reviews written by such test users, we constructed and used a default classifier trained on all the training data.

[7] Seroussi et al. (2010) chose users who had more than 50 positive and 50 negative reviews. However, in our datasets, users have fewer reviews so we set the boundary to be 5.

Table 2 shows the experimental results. Our method improves accuracies on the Blitzer and Maas datasets against the baseline classifiers. A larger improvement is achieved on the Maas dataset, probably because the average number of reviews for each product is higher than that of the Blitzer dataset such that we could estimate more reliable global features. On the Pang dataset, however, our method had no advantage. We will further analyze the reason in the following paragraphs.

On the Blitzer dataset, user leniency was more helpful than product popularity. This is probably because the Blitzer dataset includes all the reviews written by each user. On the other hand, product information plays an important role because the Maas dataset includes all the reviews for each product.

Among the two decoding methods, the easiest-first decoding consistently achieves higher accuracy. This confirms our expectation that easiest-first decoding is more cautious than two-stage decoding. However, easiest-first decoding has its own weakness in speed.

Seroussi *et al.* (2011) performed badly because the number of reviews for each user in our datasets was lower than theirs, hence, the personalized classifiers learned on limited instances would be unreliable.

Table 2 Accuracy (%) on review datasets

Method	Pang	Blitzer	Maas
Seroussi et al. (2010)	78.05	89.33	n/a
Maas et al. (2011)	**88.90**	n/a	88.89[8]
baseline	86.00	90.14	91.47
proposed (easiest-first)			
+*user*	86.00	91.03$^{\gg}$	n/a
+*product*	85.95	90.17	**92.68**
+*user* + *product*	85.55	**91.11**$^{\gg}$	n/a
proposed (two-stage)			
+*user*	85.55	90.94$^{\gg}$	n/a
+*product*	85.70	90.16	92.63
+*user* + *product*	85.50	91.02$^{\gg}$	n/a

+*user* and +*product* mean considering user leniency and product popularity features. Accuracy marked with "$\gg$" was significantly better than baseline ($p < 0.01$, assessed by McNemar's test).

[8] This result used different two-fold splitting from ours. Under their splitting, our accuracies were 90.79%, 92.39%, and 92.27% for baseline, easiest-first, and two-stage strategies, respectively. Both strategies easily beat Maas et al. (2011)'s accuracy, 88.89%. Our baseline is superior to their method, partly because of the features we used. They used only unigram features, whereas we used unigram and bigram (which considers negation) as features. With only unigram features, our baseline classifier achieved 87.80% accuracy.

4.4 Analysis

In this section, we analyze our experimental results on accuracy. We first investigate the impact on accuracy when we change the testing data size in Analysis i. Next, we show how much improvement will be gained by our method if a user (or a product) has a different number of reviews in Analysis ii. Then, we show the biases of each dataset, and the performance of our method on both emerging and existing users (and products) in Analysis iii and iv. Finally, in Analysis v and vi, we show the learning curves and some examples.

Analysis i: Impact of testing data size on speed and accuracy

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. User leniency and product popularity are both considered. We use the fixed hyper-parameters ($\phi = 1.0$, $\#\ iterations = 10$) 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 number of test reviews processed at once. Here, instead of decoding the whole testing data, we split the test reviews into equal-sized smaller subsets and apply our classifier independently to each.[9] We accumulate the results for all the subsets to compare the accuracy for the entire test data. Fig. 1 shows the experimental results. When 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 accurately capture user leniency and product popularity, which results in higher classification accuracy. We will confirm this in Analysis ii.

We then measured the testing speed using the same setting as the above experiment, while evaluating the average time consumed by one single subset. As shown in Fig. 2, the speed of the easiest-first decoding drastically slows down as the number of processed reviews grows, whereas the speed of the two-stage decoding increases linearly. Meanwhile, the accuracy of the two strategies are competitive, as shown in Fig. 1.

Based on these observations, the key factor in achieving better accuracy is not the choice of decoding strategy, but the amount of test data processed at once. We conclude that when we have too much test data for the easiest-first decoding to process in a practical time we should adopt the 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.

[9] We used the same two-fold cross-validation as the main experiment for accuracy.

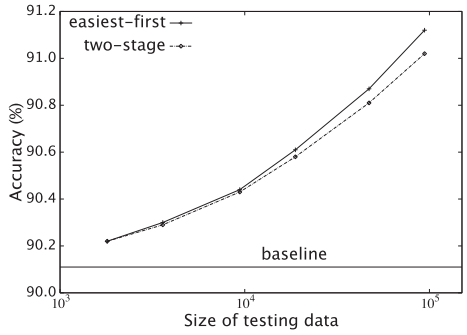

Fig. 1 Classification accuracy computed accumulatively when the number of testing reviews was changed

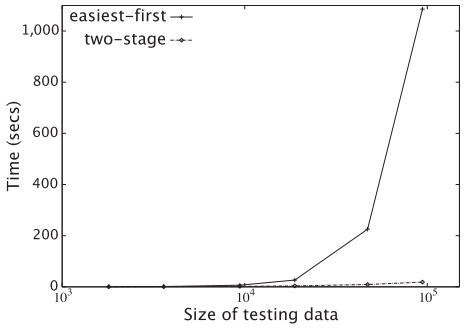

Fig. 2 Average computation time when the number of testing reviews was changed

Analysis ii: Accuracy in terms of size of neighbors

The accuracy gain is rooted in global features, while global features are computed by referring to labels of (user- and product-related) neighboring reviews, $\mathcal{S}_u(r)$ and $\mathcal{S}_p(r)$. When only one such neighbor is available, global features may be unreliable compared with those computed from many neighbors. We then investigate how the number of (user- and product-related) neighboring reviews affect accuracy improvement.

Both user leniency and product popularity features show no improvement on the Pang dataset, as in Table 3, because of the limited review size, as we illustrated in Analysis i. It is also probablly because the biases of the user and product are not sufficient in this dataset, which will be shown in Analysis iii. Table 4 shows that user leniency features greatly contribute to the improvement, while product popularity has limited influence on the Blitzer dataset. Popularity features play an important role on the Maas dataset, as shown in Table 5. In general, we 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 a larger $|\mathcal{S}_u(r)|$ or $|\mathcal{S}_p(r)|$ results in lower speed of easiest-first decoding, as shown earlier, we could bound the number of reviews written by each user or on each product to save computation without losing accuracy.

Analysis iii: Polarity bias in terms of user or product

Since our method takes advantage of biased distributions over polarity labels in terms of a user or product, larger the bias in the data, greater our method could improve classification

Table 3 Accuracy (%, lower inside cell) of proposed method (two-stage) and review size (upper inside cell) of the **Pang** dataset divided according to the number of reviews written by the user and number of reviews on the product

		No. of product-related neighbors ($\lvert\mathcal{S}_p(r)\rvert$)			
		0	1	2	3–
No. of user-related neighbors ($\lvert\mathcal{S}_u(r)\rvert$)	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)

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).

Table 4 Accuracy (%, lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the **Blitzer** dataset divided according to number of reviews written by the user and the number of reviews on the product

		No. of product-related neighbors ($\lvert\mathcal{S}_p(r)\rvert$)			
		0	1	2	3–
No. of user-related neighbors ($\lvert\mathcal{S}_u(r)\rvert$)	0	55,043 90.12 (+0.01)$^{\gg}$	34,735 90.13 (+0.27)$^{\gg}$	16,601 90.90 (+0.67)$^{\gg}$	9,630 92.37 (+0.56)$^{\gg}$
	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)$^{\gg}$	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)$^{\gg}$	4,974 91.19 (+2.37)$^{\gg}$	2,174 92.23 (+3.50)	998 89.98 (+1.70)
	8–	13,243 93.66 (+1.94)$^{\gg}$	7,484 92.34 (+1.80)$^{\gg}$	3,017 91.32 (+1.46)$^{\gg}$	1,289 90.07 (+1.55)

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).

Table 5 Accuracy (%, lower inside cell) of proposed method (two-stage) and review size (upper inside cell) on the **Maas** dataset divided according to number of reviews on the product

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

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).

accuracy. We then compute the polarity bias in terms of users (in short, *user_bias*) and 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 $\in \{+, -\}$.[10] 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 $user\_bias(u) = 0$. Higher the *user_bias* and *product_bias* values that exist, the more potential our method has to improve accuracy.

Because of 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. 3 and Fig. 4, respectively. Possibly, because the Pang dataset is collected from a discussion newsgroup, the users in it are less biased than those in the Blitzer dataset, which is collected from a general domain. As such, the user leniency features extracted from the Pang dataset might be unreliable since users do not have much bias. The products in the Maas dataset are more biased than those in the other two datasets. This could be a reason user information in the Blitzer dataset and product information in the Maas dataset contribute the most to improvement.

As illustrated in Fig. 1, when the number of reviews is small, the accuracy of our method decreases. Thus, the small size of the Pang dataset (Table 1) and low user bias values seem to

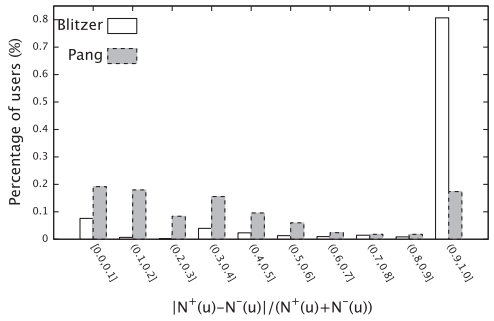

Fig. 3 User bias distribution. The users who have only one review are eliminated.

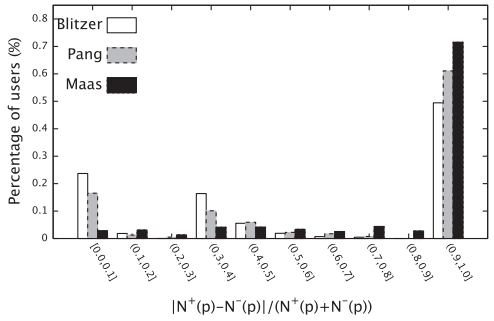

Fig. 4 Product bias distribution. The products that have only one review are eliminated.

[10] The numbers are counted using all the reviews, including training and testing data.

be the reasons our method performed badly on this dataset.

Analysis iv: Impact of training reviews written by test users (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 classification accuracy depending on whether we observed the same user (or product) in the training data. 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 estimate the labels of reviews written by seen users or on seen products better than those unseen ones' because the classifier learns n-grams specific to these users or the products (and thus is more effective in classification). On the other hand, our method performed well when more reviews were available for users and products in the test data, so we can expect consistent improvement for seen and unseen users (products).

On the Pang dataset, as shown in Table 6, the smaller number of training data resulted in poor classification accuracy, particularly on unseen users and unseen products. Unlike in the other two datasets, lack of biases and the small number of reviews seem to be responsible.

As shown in Table 7, a larger improvement was observed on reviews written by the seen users in the Blitzer dataset. We found that the average 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 in accuracy on reviews written by unseen users.

On the Maas dataset, as shown in Table 8, 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 fewer

Table 6 Accuracy (%) on seen/unseen user or product splits of **Pang** dataset

	(su, sp)	(uu, sp)	(su, up)	(uu, up)	total
No. of reviews	951	86	850	113	2,000
No. of reviews/user	3.98	1.13	3.85	1.18	4.41
No. of reviews/product	1.69	1.08	1.16	1.02	1.44
baseline	87.38	88.37	84.71	82.30	86.00
proposed (easiest-first)	87.07 (−0.32)	87.21 (−1.16)	84.24 (−0.47)	81.42 (−0.88)	85.55 (−0.45)
proposed (two-stage)	87.07 (−0.32)	87.21 (−1.16)	84.12 (−0.59)	81.42 (−0.88)	85.50 (−0.50)

su, *uu*, *sp* and *up* denote *seen user*, *unseen user*, *seen product*, and *unseen product*, respectively. The float inside parentheses is the difference between our method and baseline classifier. No accuracy is significantly different from baseline ($p \geq 0.01$, assessed by McNemar's test).

Table 7 Accuracy (%) on seen/unseen user or product splits of **Blitzer** dataset

	(su, sp)	(uu, sp)	(su, up)	(uu, up)	total
No. of reviews	35,689	60,775	36,859	55,027	188,350
No. of reviews/user	2.04	1.04	2.14	1.04	1.40
No. of reviews/product	1.20	1.39	1.14	1.20	1.43
baseline	89.72	90.33	90.50	89.94	90.14
proposed (easiest-first)	91.39 (+1.67)$^{\gg}$	90.91 (+0.58)$^{\gg}$	92.34 (+1.84)$^{\gg}$	90.33 (+0.39)$^{\gg}$	91.11 (+0.98)$^{\gg}$
proposed (two-stage)	91.24 (+1.52)$^{\gg}$	90.87 (+0.54)$^{\gg}$	92.13 (+1.63)$^{\gg}$	90.29 (+0.35)$^{\gg}$	91.02 (+0.88)$^{\gg}$

su, *uu*, *sp* and *up* denote *seen user*, *unseen user*, *seen product*, and *unseen product*, respectively. The float inside parentheses is the difference between our method and baseline classifier. Accuracy marked with "$\gg$" was significantly better than baseline ($p < 0.01$, assessed by McNemar's test).

Table 8 Accuracy (%) on seen/unseen product splits of **Maas** dataset

	(sp)	(up)	total
No. of reviews	46,397	3,603	50,000
No. of reviews/product	4.82	1.62	4.22
baseline	91.93	85.62	91.47
proposed (easiest-first)	93.07 (+1.14)	87.73 (+2.11)$^{\gg}$	92.68 (+1.21)
proposed (two-stage)	93.02 (+1.09)	87.59 (+1.97)$^{\gg}$	92.63 (+1.16)

sp and *up* denote *seen product* and *unseen product*, respectively. The float inside parentheses is the difference between our method and baseline classifier. Accuracy marked with "$\gg$" was significantly better than baseline ($p < 0.01$, assessed by McNemar's test).

reliable global features). The reason is probably that baseline classifier performed poorly on reviews written on unseen products, and hence left our method larger space for improvement.

Analysis v: Learning curves

Using the same setting as in Analysis i, we divided the training data and investigated the effect on accuracy. Fig. 5 shows the accuracy when we change the size of the training data. Our method has a clear advantage over the baseline method, even when the size of the training data is small (1,800 reviews). In other words, we do not need much training data to learn the correlation between the label and 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%).

Analysis vi: Examples

Some examples are given to explain how our model works. As shown in Table 9, our method successfully classifies some reviews that are hard to classify correctly when only textual features are used.

In the first two examples, weak negative textual features are found in the test review. How-

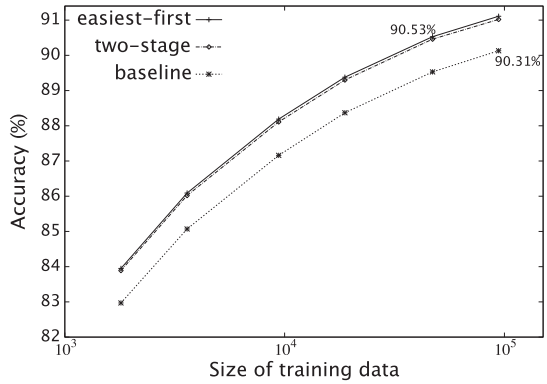

Fig. 5 Average accuracy when size of the training data is changed

Table 9 Examples show the influence of leniency and popularity global features

Leniency	Popularity	Review	Label		
			gold	baseline	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 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 helpful as 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

The **bold** content is the negative evidence learned by classifier.

ever, since the two users are lenient and product of the first review is relatively popular (these characteristics are captured by our proposed method), the 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 finds it hard to identify sentiment from words). As we can see in the third example in Table 9, the baseline classifier could recognize no obvious sentiment evidence from the textual features, while our method classified it as negative by detecting that it is about a notorious product and the user is critical.

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

5 Conclusion

We presented a method of collective sentiment classification that captures and utilizes user leniency and product popularity. Different from most of the previous studies that are aware of the user and product of the review, our model does not assume the training data to contain reviews written by the same user or on the same product in the test reviews. To determine a label 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 three real-world review datasets to compare with existing methods. The proposed method performed more accurately than the baseline, which uses only word n-grams as features when the users and products are biased on sentiment (which is often true in the real-world). It also outperformed the state-of-the-art method that combines personalized classifiers. The more reviews per user or per product are available, larger the improvement our method gains. The two-stage strategy runs in time that is linear to the number of test reviews (expected to be the same order of speed as the baseline classifiers), while achieving slightly less accuracy compared with the easiest-first strategy.

We consider our proposed method as a first step toward modeling more complex properties of reviews. A future extension of this work is to detect a user's preference for a certain kind of product. We also plan to use dual decomposition (Koo, Rush, Collins, Jaakkola, and Sontag 2010) as an advanced decoding strategy for our collective sentiment classification.

Acknowledgement

The authors would like to thank the reviewers for their valuable comments. This work is based on the conference paper (Gao, Yoshinaga, Kaji, and Kitsuregawa 2013) with extended experiment and detailed result analysis.

Reference

Blitzer, J., Dredze, M., and Pereira, F. (2007). "Biographies, Bollywood, Boom-boxes and Blenders: Domain Adaptation for Sentiment Classification." In *Proceedings of ACL*, pp. 440–447, Prague, Czech Republic.

Dredze, M., Crammer, K., and Pereira, F. (2008). "Confidence-weighted Linear Classification." In *Proceedings of ICML*, pp. 264–271, Helsinki, Finland.

Gao, W., Yoshinaga, N., Kaji, N., and Kitsuregawa, M. (2013). "Collective Sentiment Classification Based on User Leniency and Product Popularity." In *Proceedings of Paclic*, pp. 357–365, Taipei, Taiwan.

Hu, M. and Liu, B. (2004). "Mining and Summarizing Customer Reviews." In *Proceedings of KDD*, pp. 168–177, Seattle, WA, USA.

Koo, T., Rush, A. M., Collins, M., Jaakkola, T., and Sontag, D. (2010). "Dual Decomposition for Parsing with Non-projective Head Automata." In *Proceedings of EMNLP*, pp. 1288–1298, Cambridge, MA, USA.

Krishnan, V. and Manning, C. D. (2006). "An Effective Two-stage Model for Exploiting Non-local Dependencies in Named Entity Recognition." In *Proceedings of COLING-ACL*, pp. 1121–1128, Sydney, NSW, Australia.

Li, F., Liu, N., Jin, H., Zhao, K., Yang, Q., and Zhu, X. (2011). "Incorporating Reviewer and Product Information for Review Rating Prediction." In *Proceedings of IJCAI*, pp. 1820–1825, Barcelona, Spain.

Li, S., Lee, S. Y. M., Chen, Y., Huang, C.-R., and Zhou, G. (2010). "Sentiment Classification and Polarity Shifting." In *Proceedings of COLING*, pp. 635–643, Beijing, China.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., and Potts, C. (2011). "Learning Word Vectors for Sentiment Analysis." In *Proceedings of ACL-HLT*, pp. 142–150, Portland, Oregon, USA.

Nakagawa, T., Inui, K., and Kurohashi, S. (2010). "Dependency Tree-based Sentiment Classification Using CRFs with Hidden variables." In *Proceedings of NAACL-HLT*, pp. 786–794, Los Angeles, CA, USA.

Pang, B. and Lee, L. (2004). "A Sentimental Education: Sentiment Analysis using Subjectivity Summarization Based on Minimum Cuts." In *Proceedings of ACL*, pp. 271–278, Barcelona, Spain.

Pang, B. and Lee, L. (2008). "Opinion Mining and Sentiment Analysis." *Foundation and Trends in Information Retrieval*, **2** (1-2), pp. 1–135.

Pang, B., Lee, L., and Vaithyanathan, S. (2002). "Thumbs Up? Sentiment Classification Using Machine Learning Techniques." In *Proceedings of EMNLP*, pp. 79–86, Pennsylvania, PA, USA.

Qiu, G., Liu, B., Bu, J., and Chen, C. (2011). "Opinion Word Expansion and Target Extraction through Double Propagation." *Computational Linguistics*, **37** (1), pp. 9–27.

Seroussi, Y., Zukerman, I., and Bohnert, F. (2010). "Collaborative Inference of Sentiments from Texts." In *Proceedings of UMAP*, pp. 195–206, Big Island, HI, USA.

Socher, R., Pennington, J., Huang, E. H., Ng, A. Y., and Manning, C. D. (2011). "Semi-Supervised Recursive Autoencoders for Predicting Sentiment Distributions." In *Proceedings of EMNLP*, pp. 151–161, Edinburgh, Scotland, UK.

Speriosu, M., Sudan, N., Upadhyay, S., and Baldridge, J. (2011). "Twitter Polarity Classification with Label Propagation over Lexical Links and the Follower Graph." In *Proceedings of EMNLP, Workshop on Unsupervised Learning in NLP*, pp. 53–63, Edinburgh, UK.

Tan, C., Lee, L., Tang, J., Jiang, L., Zhou, M., and Li, P. (2011). "User-level Sentiment Analysis Incorporating Social Networks." In *Proceedings of KDD*, pp. 1397–1405, San Diego, California, USA.

Tsuruoka, Y. and Tsujii, J. (2005). "Bidirectional Inference with the Easiest-first Strategy for Tagging Sequence Data." In *Proceedings of HLT-EMNLP*, pp. 467–474, Vancouver, B.C., Canada.

Turney, P. D. (2002). "Thumbs Up or Thumbs Down?: Semantic Orientation Applied to Unsupervised Classification of Reviews." In *Proceedings of ACL*, pp. 417–424, Pennsylvania, PA, USA.

Wenliang Gao: He received his B.Sc. in Software Engineering and M.Sc. in Computer Science and Technology from the Dalian University of Technology, China in 2007 and 2009, respectively. He is currently pursuing his Ph.D. at the Graduate School of Information Science and Technology, the University of Tokyo, Japan. His research interests include natural language processing and machine learning.

Nobuhiro Kaji: He received his Ph.D. in Information Science and Technology from the University of Tokyo, Japan, in 2005. He worked at the Institute of Industrial Science, the University of Tokyo, as a Research Associate and Project Assistant Professor from 2005 to 2006 and from 2006 to 2012, respectively. He is currently a Project Associate Professor at the Institute of Industrial Science, the University of Tokyo, and a Senior Researcher at the National Institute of Information and Communications Technology (NICT), Japan. His research interests include natural language processing and machine learning.

Naoki Yoshinaga: He received his B.Sc. and M.Sc. in Information Science and Ph.D. in Information Science and Technology from the University of Tokyo, Japan in 2000, 2002, and 2005, respectively. He was a JSPS research fellow

from 2002 to 2005 (DC1) and from 2005 to 2008 (PD). He worked at the Institute of Industrial Science, the University of Tokyo, as a Project Researcher and a Project Assistant Professor from 2008 to 2012. He is currently a Project Associate Professor at the Institute of Industrial Science, the University of Tokyo, and a Senior Researcher at the National Institute of Information and Communications Technology (NICT), Japan. His research interests include computational linguistics and machine learning.

Masaru Kitsuregawa: He received his Ph.D. in Information Engineering from the University of Tokyo, Japan, in 1983. He is currently a Professor at the Institute of Industrial Science, the University of Tokyo, Director General/Professor at Earth Observation Data Integration & Fusion Research Initiative of the University of Tokyo, Director General of the National Institute of Informatics, and President of Information Processing Society of Japan. His research interests include high performance database engineering and big data systems. He is a recipient of the ACM SIGMOD Edgar F. Codd Innovations Award (2009), and a co-recipient of the IPSJ Contribution Award (2010) and the Medal with Purple Ribbon (2013).

(Received November 1, 2013)
(Revised January 9, 2014)
(Accepted February 28, 2014)