Application of Semi-supervised Learning to Evaluative Expression Classification

Yasuhiro Suzuki[1,*], Hiroya Takamura[2], and Manabu Okumura[2]

[1] Interdisciplinary Graduate School of Science and Engineering,
Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku Yokohama, Japan, 226-8503
`yasu@lr.pi.titech.ac.jp`
[2] Precision and Intelligence Laboratory, Tokyo Institute of Technology,
4259 Nagatsuta Midori-ku Yokohama, Japan, 226-8503
{`takamura, oku`}`@pi.titech.ac.jp`

Abstract. We propose to use semi-supervised learning methods to classify evaluative expressions, that is, tuples of subjects, their attributes, and evaluative words, that indicate either favorable or unfavorable opinions towards a specific subject. Due to its characteristics, the semi-supervised method that we use can classify evaluative expressions in a corpus by their polarities. This can be accomplished starting from a very small set of seed training examples and using contextual information in the sentences to which the expressions belong. Our experimental results with actual Weblog data show that this bootstrapping approach can improve the accuracy of methods for classifying favorable and unfavorable opinions.

1   Introduction

An increasing amount of work has been devoted to investigating methods of detecting favorable or unfavorable opinions towards specific subjects (e.g., companies and their products) within online documents such as Weblogs (blogs), messages in a chat room and on bulletin board (BBS) [1, 2, 7, 9, 11, 12, 18]. Areas of application for such an analysis are numerous and varied, ranging from analysis of public opinion, customer feedback, and marketing analysis to detection of unfavorable rumors for risk management. The analyses are potentially useful tools for the commercial activities of both companies and individual consumers who want to know the opinions scattered on the World Wide Web (WWW).

To analyze a huge amount of favorable or unfavorable opinions, we need to automatically detect evaluative expressions in text.

Evaluative expressions are not mere words that indicate unique (favorable or unfavorable) polarity in themselves (such as the adjectives 'beautiful' and 'bad'), but rather they are tuples of the subject to be evaluated, an attribute, and an evaluative word. Tuples are necessary because the evaluative polarity of

---

* Yasuhiro Suzuki currently works at Fujitsu.

an individual word is often ambiguous and is determined context-dependently. In the following example [1], which is rather artificial due to direct translation, the first sentence is positive and the second sentence is negative, although both have the same word "high".

– The storage capacity of this HDD is high.
– The noise of this HDD is high.

We thus define an evaluative expression as a tuple of a subject, an attribute, and an evaluative word (in the above example, 'HDD', 'capacity'/'noise', and 'high').

A good way to automatically acquire evaluative expressions is to first extract candidate expressions from a collection of documents and to automatically classify them as positive (favorable), negative (unfavorable), or neutral (non-evaluative). Therefore, we study classifying candidate evaluative expressions in documents into the three classes.

Much work has been done to automatically label a piece of text according to its positive or negative polarity, as detailed in the next section. Several previous papers have addressed the task by building classifiers that rely exclusively upon labeled examples [1, 2]. By training classifiers from labeled data, one can apply familiar and powerful machine learning techniques such as SVM (Support Vector Machines) [3] and the naive Bayes classifier. However, in practice, obtaining enough labeled examples to train the classifiers accurately may be difficult.

A contrasting approach relies upon only unlabeled data. This makes using a large corpus possible. However, a drawback to such an approach is its low accuracy when used on actual data.

In the area of machine learning, using labeled and unlabeled examples together has often been found effective [4]. In this paper, therefore, we explore the use of bootstrapping methods that allow evaluative classifiers to be trained on a collection of unlabeled texts. Using labeled examples as training data, we apply a supervised learning algorithm to automatically train the classifier. The trained classifier can be then used to automatically classify evaluative expressions into polarity categories and generate more labeled examples, which in turn increases the training data, and this entire process can be repeated.

To implement the idea, in this work, we adopted the EM algorithm [5] for the bootstrapping method and the naive Bayes classifier for the supervised learning method. Specifically, using various contextual information (detailed in Sec. 4.2) as features in the classifier trained with small seed training examples, evaluative expressions and their contextual information are newly identified in unannotated texts. Trained again with the labeled examples, our classifier identifies more evaluative expressions in unannotated texts. We adopted the combination of the EM algorithm and the naive Bayesian method, because Nigam et al. [4] have already shown that this combination can yield better performance in the text classification task.

---

[1] Though we use Japanese text data, we illustrate our method with English examples for better understanding by non-Japanese readers.

2   Related Work

The previous work on sentiment classification can be divided into the following three classes, according to the target unit of text: a word (or expression), a sentence (or clause), or a full document.

- Word sentiment classification [6, 7, 8, 9]
- Sentence sentiment classification [10, 11]
- Document sentiment classification [1, 12, 2]

From another viewpoint, the work on sentiment classification can be divided into two different approaches: those that try to learn classifiers directly from the training corpus (supervised learning method) [1, 2, 10, 11, 12] and those targeting the acquisition of evaluative lexica in an unsupervised fashion [6, 7, 8, 9].

In the latest studies on document sentiment classification, classifiers based on machine learning (e.g., [1, 2]) performed better than knowledge-intensive classifiers. One of the main obstacles to producing a sentiment classifier in a supervised fashion is a lack of training data. Targeting words (expressions) makes obtaining training data more difficult because the data must be manually annotated with polarities. Because manually producing annotated data is time consuming, the amount of available annotated data is relatively small[2].

As mentioned in the Introduction, much work has been done on semi-supervised learning methods [4, 14, 17]. Since unannotated texts are easy to obtain, the semi-supervised learning framework can produce a much larger collection of labeled examples than are currently available in manually created data. Consequently, we believe that sentiment classification systems can be trained on extremely large text collections by applying this framework.

We think the work of Riloff and Wiebe [13] is most relevant to ours because they also used a semi-supervised learning method. Their classifier targeted sentences, they tried to learn extraction patterns for subjective (evaluative) expressions from the training data, These extraction patterns were then used for the classification. Roughly, their work was the application of a semi-supervised learning technique to subjective expressions, which is similar to our work. However, their work concentrated on the classification of sentences as subjective or objective, while our classification targets evaluative expressions as positive, negative, or neutral (non-evaluative).

Furthermore, while Riloff and Wiebe used only extracted patterns for subjective expressions to classify sentences, we use a variety of contextual information that can be obtained from the sentence to which an evaluative expression belongs.

3   Our Method

We propose to use a semi-supervised learning method for classifying evaluative expressions (i.e., tuples of a subject, an attribute and an evaluative word) into

---

[2] Targeting full documents is easier, because more training data, in the form of reviews, can be found on the WWW.

three classes: positive, negative or neutral[3]. We suppose that evaluative expressions appear with certain types of context, such as 'I am really happy, because the storage capacity is high.' or 'Unfortunately, the laptop was too expensive.' We would like to extract such contexts from labeled examples and then use those contexts to re-label examples. By iterating this procedure, we would be able to accurately classify evaluative expressions and simultaneously collect evaluative words and typical contexts.

In order to achieve this bootstrapping, we used the EM algorithm [5], which has a theoretical base of likelihood maximization of incomplete data and can enhance supervised learning methods. We specifically adopted the combination of the naive Bayes classifiers and the EM algorithm. This combination has been proven to be effective in the text classification [4]. Another famous semi-supervised method that has been shown to be effective in text classification is co-training [14]. We however could not use co-training in this task, since we do not have conditionally independent views, which are required for co-training.

We explain the EM-based method in the following section.

3.1   Evaluative Expression Classification with Naive Bayes Classifiers

This model has been successfully applied to text categorization and its generative probability of example $\mathbf{x}$ given a category c has the form :

$$P(\mathbf{x}|c, \theta) = P(|\mathbf{x}|)|\mathbf{x}|! \prod_w \frac{P(w|c)^{N(w,\mathbf{x})}}{N(w,\mathbf{x})!}, \qquad (1)$$

where $P(|\mathbf{x}|)$ denotes the probability that a text of length $|\mathbf{x}|$ occurs, $N(w,\mathbf{x})$ denotes the number of occurrences of w in text $\mathbf{x}$, and θ denotes all the parameters of the model. The occurrence of a text is modeled as a set of events, in which a word is drawn from the whole vocabulary.

In evaluative expression classification, categories c are the positive category, the negative category and the neutral category. Instances $\mathbf{x}$ are represented by features including evaluative words and their context. A detailed description of features will be given in Sec. 4.

3.2   Incorporation of Unlabeled Data with the EM Algorithm

The EM algorithm is a method to estimate a model that has the maximal likelihood of the data when some variables cannot be observed (these variables are called *latent variables*) [5]. Nigam et al. [4] proposed a combination of the naive Bayes classifiers and the EM algorithm, which we also use as a base for constructing a Fisher kernel.

---

[3] Here, 'evaluative expressions' are actually candidates of evaluative expressions. Non-evaluative expressions are classified as neutral.

Ignoring the unrelated factors of Eq. (1), we obtain

$$P(\mathbf{x}|c,\theta) \propto \prod_w P(w|c)^{N(w,\mathbf{X})}, \quad P(\mathbf{x}|\theta) \propto \sum_c P(c) \prod_w P(w|c)^{N(w,\mathbf{X})}. \quad (2)$$

If we regard c as a latent variable and introduce a Dirichlet distribution as the prior distribution for the parameters, the Q-function (i.e., the expected log-likelihood) of this model is defined as :

$$Q(\theta|\bar{\theta}) = \log(P(\theta)) + \sum_{\mathbf{X} \in D} \sum_c P(c|\mathbf{x},\bar{\theta}) \log\left(P(c) \prod_w P(w|c)^{N(w,\mathbf{X})} \right), \quad (3)$$

where $P(\theta) \propto \prod_c (P(c)^{\alpha-1} \prod_w (P(w|c)^{\alpha-1}))$; a Dirichlet distribution. α is a user-given parameter and D is the set of examples used for model estimation.

Instead of the usual EM algorithm, we use the tempered EM algorithm [15], and obtain the following EM steps :
E-step:

$$P(c|\mathbf{x},\bar{\theta}) = \frac{\left(P(c|\bar{\theta})P(\mathbf{x}|c,\bar{\theta})\right)^{\beta}}{\sum_c \left(P(c|\bar{\theta})P(\mathbf{x}|c,\bar{\theta})\right)^{\beta}}, \quad (4)$$

M-step:

$$P(c) = \frac{g(\alpha,\bar{\theta},c)}{\sum_c g(\alpha,\bar{\theta},c)}, \quad P(w|c) = \frac{h(\alpha,\bar{\theta},w,c)}{\sum_w h(\alpha,\bar{\theta},w,c)}, \quad (5)$$

where

$$g(\alpha,\bar{\theta},c) = (\alpha - 1) + \sum_{\mathbf{X} \in D} P(c|\mathbf{x},\bar{\theta}), \quad (6)$$

$$h(\alpha,\bar{\theta},w,c) = (\alpha - 1) + \sum_{\mathbf{X} \in D} P(c|\mathbf{x},\bar{\theta})N(w,\mathbf{x}). \quad (7)$$

For labeled example $\mathbf{x}$, Eq. (4) is not used. Instead, $P(c|\mathbf{x},\bar{\theta})$ is set as 1.0 if c is the category of $\mathbf{x}$, otherwise 0.

As can be seen from Eq. (5), the larger α is, the more uniform the distribution becomes. In practice, α is treated as a user-given parameter. By decreasing hyper-parameter β, we can reduce the influence of intermediate classification results if those results are unreliable.

Too much influence by unlabeled data sometimes deteriorates the model estimation. Therefore, we introduce a new hyper-parameter λ (≥ 0.0), which acts as weight on unlabeled data [4]. In the second term on the right-hand-side of Eq. (3), unlabeled training examples in D are weighted by λ. We can reduce the influence of unlabeled data by decreasing the value of λ.

We derived new update rules from this new Q-function. The EM computation stops if the difference in values of the Q-function is smaller than a threshold.

3.3   Hyper-Parameter Prediction

Classification results depend largely on two hyper-parameters, specifically λ and β. We would like to predict good values of λ and β. The simplest methods are leave-one-out estimation or cross-validation. However, those methods require a high computational cost, especially when we use an EM-like iterative algorithm. Therefore, we propose an efficient quasi-leave-one-out estimation method.

Our method evaluates the accuracy for classifying labeled training examples. For each training example, we add minimal modification to the estimated parameters (excluding hyper-parameters) so that we can obtain new parameters $P_k(c)$ and $P_k(w|c)$ that are estimated without using the example. Formally we use the following parameters for training example $\mathbf{x}_k$:

$$P_k(c) = \frac{g(\alpha, \bar{\theta}, c) - P(c|\mathbf{x}_k, \bar{\theta})}{\sum_c \left(g(\alpha, \bar{\theta}, c) - P(c|\mathbf{x}_k, \bar{\theta}) \right)}, \tag{8}$$

$$P_k(w|c) = \frac{h(\alpha, \bar{\theta}, w, c) - P(c|\mathbf{x}_k, \bar{\theta})N(w, \mathbf{x}_k)}{\sum_w \left(h(\alpha, \bar{\theta}, w, c) - P(c|\mathbf{x}_k, \bar{\theta})N(w, \mathbf{x}_k) \right)}. \tag{9}$$

Thus, by preserving the values of functions $g(\alpha, \bar{\theta}, c)$ and $h(\alpha, \bar{\theta}, w, c)$, we can efficiently compute the modified parameters for each labeled training example. Henceforth, we calculate the quasi-leave-one-out accuracy. We select the hyper-parameters that yield the best quasi-leave-one-out accuracy. Please notice that all the labeled training examples are used in EM iterations and therefore this procedure is not an actual leave-one-out, but a quasi-leave-one-out.

3.4   Fisher Kernel (Fisher Score)

The Fisher kernel [16] is a similarity function, which is actually the dot-product of two Fisher scores. The Fisher score of an example is obtained by partially differentiating the log-likelihood of the example with respect to parameters. The Fisher score indicates approximately how the probability model will change if the example is added to the training data that is used in the estimation of the model. That means, the Fisher kernel between two samples will be large, if the influences of the two samples are similar and large. Takamura and Okumura reported that the Fisher kernel based on a probability model estimated by the semi-supervised EM algorithm works well in text categorization [17]. One good thing about the combination of the Fisher kernel and the EM algorithm is that high-performance kernel classifiers such as SVMs can be used in a somewhat semi-supervised way. We constructed the Fisher kernel on the basis of the above EM-estimated model described above as proposed by Takamura and Okumura [17]. Please refer to their paper for a detailed explanation.

4   Data Preparation and Features

4.1   Data Preparation

As the data for the experiments, we use real Weblog (blog) data collected by the system called blogWatcher [18]. From the blog data, we obtained candidate

evaluative expressions and contextual information in the sentences to which the expressions belong, by segmenting HTML documents into sentences and applying a Japanese syntactic analyzer to the sentences to yield their syntactic structures. Hereafter, we call a pair of a candidate expression and its contextual information an example. The reason why we adopted blogs as our data source is that they contain more evaluative expressions, and they are easier to collect, than the newspaper corpora usually used in NLP research. We used as the Japanese syntactic analyzer Cabocha[4]. Sentence boundaries were detected in a heuristic way.

Then, from the sentences, candidate evaluative expressions, that is, tuples of subjects, their attributes, and evaluative words, are extracted. We extract candidate expressions only in cases where evaluative words are adjectives. For each adjective, we try to find the nouns for a subject and an attribute. If the nouns that modify the adjective in the syntactic structure satisfy some restrictions[5], they are extracted as the nouns for the subject and the attribute. The actual phenomena of evaluations in text are more complicated than these tuples, as was discussed by Wiebe [20]. However, we believe that this tuple-based definition of evaluative expressions will give a good approximation of the actual phenomena.

By randomly sampling 200 expressions, we evaluated our method's effectiveness for extracting candidate expressions, and found that it yielded an accuracy of 64%. Therefore, we consider that some percentages of the errors in the experiments was caused by the naiveness of our method of extracting candidate expressions.

4.2   Contextual Information Used for Classification

In Sec. 3, we explained that we adopted the naive Bayes classifier. In this subsection, we describe various types of contextual information that are used as features in the classifier. Contextual information can be extracted from the sentence to which the corresponding candidate evaluative expression belongs.

We assume that evaluative expressions are accompanied by various kinds of information that are useful for deciding their polarities. For example, if we already know that 'good' is a positive expression, from a sentence 'Good, since the storage capacity of the laptop is high', we can determine that 'capacity is high' is a positive expression. We can conjecture that a causal conjunction tends to connect expressions in the same polarity. Using the knowledge, if there is a type of sentence 'A, since B', we can determine B's polarity from A's, and vice versa.

Thus, in this work we take into account the following contextual information for an candidate evaluative expression:

1. Candidate evaluative expression itself
2. Exclamation words detected by a part-of-speech tagger
3. Emoticons in the sentence and their emotional categories

---

[4] It is available at `http://chasen.org/~taku/software/cabocha`.
[5] Roughly, the subjects should be concrete nouns, and the attributes should be abstract nouns in our thesaurus [19].

4. Words that modify the words in the tuples (candidate expressions)
5. Word that is modified by the candidate evaluative word
6. Words that are in the same 'bunsetsu' as the candidate evaluative word[6]

Emoticons can be considered as useful, since smileys tend to cooccur with positive expressions and sad faces tend to cooccur with negative expressions. Emoticons are automatically extracted from a sentence and classified into the six categories (happy, sad, angry, surprised, acting, and forced smile), using the method discussed in the work of Tanaka et al.'s [21].

A negation word 'not'[7] reverses the polarity of an evaluative word just before it. Therefore, taking into account this characteristic, if a candidate expression is followed by the negation word, the combination of the expression and the negation word is treated as a feature. Specifically, 'not bad' is treated as 'bad' + odd number of negations, 'not not bad' is treated as 'bad' + even number of negations, respectively. This definition of the scope of negation words should be discussed further in future work. Parsing results will provide us with good clues for that purpose. We will also have to collect other negation words, though we just use 'not' in this work.

Similarly, if a candidate expression modifies or is modified by any evaluative expression with a contrastive or adversative conjunction, the polarities of those expressions are poles apart. Therefore, in these cases, the feature 'reverse' is added to the contextual information.

Consider, for example, a sentence belonging to negative : 'Phew, the noise of this HDD is annoyingly high :-('. In the sentence, we can find a tuple of subject 'HDD', attribute 'noise', and evaluative word 'high'. For the tuple, we can extract the following contextual information as features: the tuple itself, an exclamation 'phew', a modifying word 'annoyingly' and an emoticon ':-('.

4.3   Statistics of the Data

As mentioned in Sec. 4.1, since text data on the web is noisy and our preprocessing module that uses publicly available Japanese morphological and syntactic analyzers sometimes makes errors, the data for our experiments is rather noisy. Therefore, we use the following heuristics to filter the examples that may be considered to contain errors :

– No contextual information can be obtained,
– neither subject nor attribute are extracted,
– the distance between the evaluative word and the subject and/or the attribute is more than 16 bytes[8].

Furthermore, since the features that seldom appear are considered to be ineffective, we only used those features that appeared more than twice. Approximately

---

[6] A 'bunsetsu' is a unit in Japanese that consists of a content word (noun, verb, adjective) and some closed words (postposition, auxiliary verb).

[7] In the experiment, a Japanese negation word 'nai' is regarded as a negation word instead of 'not', since our dataset is in Japanese.

[8] Examples with arge distances often contain errors.

2.6 million examples were extracted from a blog collection. We obtained 35,765 examples after the filtering. Although many examples were filtered out, if a good syntactic parser trained for rather noisy text such as web documents becomes available in the near future, we would be able to use more examples.

Then, we manually labeled a subset of the examples, to use them as either training data or as test data for the evaluation. The subset were labeled as belonging to one of the following classes: neutral (non-evaluative), positive evaluation, and negative evaluation. We labeled 1,061 examples, and the proportion of the labels is as follows: neutral (69; 6.5%), positive (504; 47.5%), and negative (488; 46.0%). To check the reliability of the annotation, we compared the annotation results of two annotators. The rate of inter-annotator agreement was 91.5%.

5   Experiments

We use the 1061 labeled examples for evaluation. The number of unlabeled training examples was 34704.

As an evaluation measure, we used *accuracy*, which is defined as the number of the correctly classified examples divided by the total number of the examples. The baseline accuracy was 47.5%, which is the ratio of the examples belonging to the positive evaluation class in the 1061 labeled examples.

We conducted experiments for different values of hyper-parameters : 0.0005 to 1.0 for λ and 0.001 to 1.0 for β. We used the hyper-parameter prediction method introduced in Sec. 3.3. The user-given parameter α for the naive Bayes classifiers was fixed to 2.0. As for SVM classification, we conducted experiments with several different values of the soft-margin parameter C, and selected the value that produced the best accuracy.

5.1   Results

Comparison of methods
Table 1 shows the accuracy values for the various methods. Incorporation of unlabeled data improves classification accuracy of the naive Bayes classifiers for this task. The Fisher kernel on the probability model estimated with a semi-supervised method, which is referred to as SVM+NaiveBayes+EM in the table, also improves SVM performance.

If the actual best values of β and λ are selected for each fold of cross-validation, the accuracy reaches 79.5%. Although this is unfair, brushing up hyper-parameter selection would further improve the method's accuracy.

Though the actual best values were not selected, the proposed method for hyper-parameter prediction also worked well.

Influence of labeled training data size
The accuracy values for different sizes of labeled training data are presented in Figure 1. The values were obtained through 10, 5, 3, 2-fold cross validations and inverted cross-validations that match up the training/test dataset sizes. In

Table 1. Accuracy for each method; "NB" corresponds to the naive Bayes classifier, "NB+EM" corresponds to the naive Bayes classifier enhanced with EM, and "SVM+NB+EM" corresponds to the SVM that uses the Fisher kernel extracted from NB+EM model

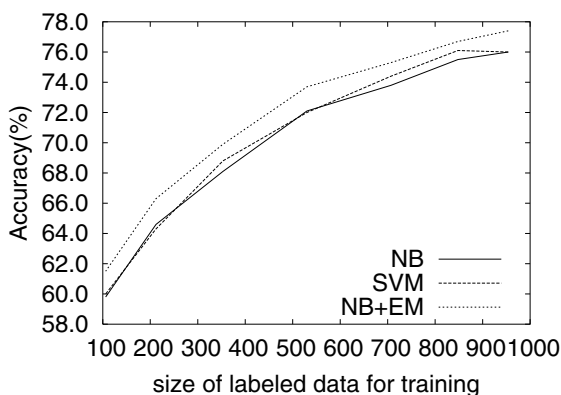| Method	Accuracy(%)
Baseline	47.5
NB	76.0
SVM	76.6
NB+EM	77.1
SVM+NB+EM	77.9

Fig. 1. Accuracy vs Labeled Training Data Size; "NB" corresponds to the result of classification with the naive Bayes classifier, and "NB+EM" corresponds to the result of classification with the naive Bayes classifier enhanced with EM

inverted cross-validations, the smaller of the two split datasets was used for training. The best values for β and λ were used in this experiment.

This result shows that our semi-supervised EM algorithm boosted accuracy, regardless of the size of labeled training data. The difference in the accuracy before and after the EM computation was statistically significant in the sign-test with a 5% significance-level.

Influence of unlabeled training data size

The accuracy values for different sizes of unlabeled training data are given in Figure 2. This result shows that even a relatively small sized unlabeled dataset (e.g., 5000 examples) improved the accuracy value. As this curve shows, although only approximately 35,000 unlabeled examples are currently available, we can expect better accuracy for a larger unlabeled training dataset.

Many of the classification errors were caused by errors in dependency analysis and failure to detect subjects and attributes. The existing dependency parsers are designed for well-formatted text such as newspaper articles, not for Web
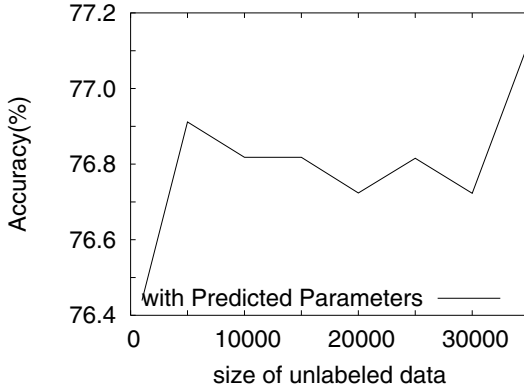
Fig. 2. Accuracy of NB+EM vs Unlabeled Data Size; note that the range for the y-axis is different from the previous figure

documents. Improvement in parsing technology would solve this problem. We currently rely on some heuristics to detect subjects and attributes. We require more sophisticated detection methods to avoid such errors.

Some errors were related to limitations of the proposed method. For example, our method still has difficulty dealing with idiomatic expressions or ambiguous words. We need to extend the method so that combinations of multiple features (words) are taken into consideration.

In order to qualitatively analyze the features, we extracted the 100 features that had the largest $P(w|positive)$ before and after EM computation. Compared with the top 100 features before EM, more contextual features were found after EM, such as, exclamations, the facemark (emoticon) category *happy*, a negation word + 'but', therefore + 'interesting', therefore + 'comfortable'.

6   Conclusions

We proposed to use a semi-supervised method for automatically classifying evaluative expressions as positive, negative, or neutral. We adopted the EM algorithm and the naive Bayes classifiers together with a method for predicting hyper-parameters. We also used the Fisher kernel on the model that we estimated with the semi-supervised method. We empirically demonstrated that the semi-supervised method works well for classifying the evaluative expressions.

References

1. Pang, B., Lee, L., Vaithyanathan, S.: Thumbs up? sentiment classification using machine learning techniques. EMNLP'02. (2002) 76–86
2. Dave, K., Lawrence, S., Pennock, D.M.: Mining the peanut gallery: Opinion extraction and semantic classification of product reviews. 12th WWW Conference. (2003) 519–528

3. Cristianini, N., and Shawe-Taylor, J.: An Introduction to Support Vector Machines (and other kernel-based learning methods), Cambridge University Press, (2000)
4. Nigam, K., McCallum, A., Thrun, S., Mitchell, T.: Text classification from labeled and unlabeled documents using EM. Machine Learning **39** (2000) 103–134
5. Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society Series B **39** (1977) 1–38
6. Hatzivassiloglou, V., McKeown, K.R.: Predicting the semantic orientation of adjectives. 35th ACL. (1997) 174–181
7. Turney, P.D.: Thumbs up? thumbs down? semantic orientation applied to unsupervised classification of reviews. 40th ACL. (2002) 417–424
8. Kamps, J., Marx, M., Mokken, R.J., de Rijke, M.: Using wordnet to measure semantic orientations of adjectives. 4th LREC. (2004) 1115–1118
9. Kim, S.M., Hovy, E.: Determining the sentiment of opinions. 20th COLING. (2004) 1367–1373
10. Kudo, T., Matsumoto, Y.: A boosting algorithm for classification of semi-structured text. EMNLP'04. (2004) 301–308
11. Wilson, T., Wiebe, J., Hwa, R.: Just how mad are you? finding strong and weak opinion clauses. 19th AAAI. (2004)
12. Pang, B., Lee, L.: A sentimental education: Sentiment analysis using subjectivity summarization based on minimum cuts. 42th ACL. (2004) 271–278
13. Riloff, E., Wiebe, J.: Learning extraction patterns for subjective expressions. EMNLP'03. (2003) 105–112
14. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. Proceedings of the Workshop on Computational Learning Theory. (1998) 92–100
15. Hofmann, T., Puzicha, J.: Statistical models for co-occurrence data. Technical Report AIM-1625, Artifical Intelligence Laboratory, Massachusetts Institute of Technology (1998)
16. Jaakkola, T., Haussler, D.: Exploiting generative models in discriminative classifiers. NIPS 11. (1998) 487–493
17. Takamura, H., Okumura, M.: A comparative study on the use of labeled and unlabeled data for large margin classifiers. 1st IJCNLP2004. (2004) 620–625
18. Nanno, T., Fujiki, T., Suzuki, Y., Okumura, M.: Automatically collecting, monitoring, and mining japanese weblogs. 13th WWW Conference. (2004) 320–321
19. Ikehara, S., Miyazaki, M., Shirai, S., Yokoo, A., Nakaiwa, H., Ogura, K., Ooyama, Y., Hayashi, Y.: Goi-Taikei – A Japanese Lexicon. Iwanami Shoten (1997)
20. Wiebe, J.: Instructions for annotating opinions in newspaper articles. Technical report, University of Pittsburgh Technical Report (TR-02-101) (2002)
21. Tanaka, Y., Takamura, H., Okumura, M.: Extraction and classification of facemarks with kernel methods. IUI 2005. (2005) 28–34