Sentiment Classification Using Word Sub-sequences and Dependency Sub-trees

Shotaro Matsumoto, Hiroya Takamura, and Manabu Okumura

Tokyo Institute of Technology, Precision and Intelligence Laboratory, 4259 Nagatsuta-cho Midori-ku Yokohama, Japan
shotaro@lr.pi.titech.ac.jp
{takamura, oku}@pi.titech.ac.jp

Abstract. Document sentiment classification is a task to classify a document according to the positive or negative polarity of its opinion (favorable or unfavorable). We propose using syntactic relations between words in sentences for document sentiment classification. Specifically, we use text mining techniques to extract frequent word sub-sequences and dependency sub-trees from sentences in a document dataset and use them as features of support vector machines. In experiments on movie review datasets, our classifiers obtained the best results yet published using these data.

1 Introduction

There is a great demand for information retrieval systems which are able to handle reputations behind documents such as customer reviews of products on the web. Since sentiment analysis technologies which identify sentimental aspects of a text are necessary for such a system, the number of researches for them has been increasing. As one of the problems of sentiment analysis, there is a document sentiment classification task to label a document according to the positive or negative polarity of its opinion (favorable or unfavorable). A system using document sentiment classification technology can provide quantitive reputation information about a product as the number of positive or negative opinions on the web.

In the latest studies on document sentiment classification, classifiers based on machine learning (e.g., [1], [6], [10]), which have been successful in other document classification tasks, showed higher performance than rule-based classifiers. Pang et al. [2] reported 87% accuracy rate of document sentiment classification of the movie reviews by their classifier using word unigram as feature for support vector machines (SVMs).

For these classifiers, a document is represented as a bag-of-words, where a text is regarded as a set of words. Therefore, the document representation ignores word order and syntactic relations between words appearing in a sentence included in the original document. However, not only a bag-of-words but also word order and syntactic relations between words in a sentence are intuitively

T.B. Ho, D. Cheung, and H. Liu (Eds.): PAKDD 2005, LNAI 3518, pp. 301–311, 2005.

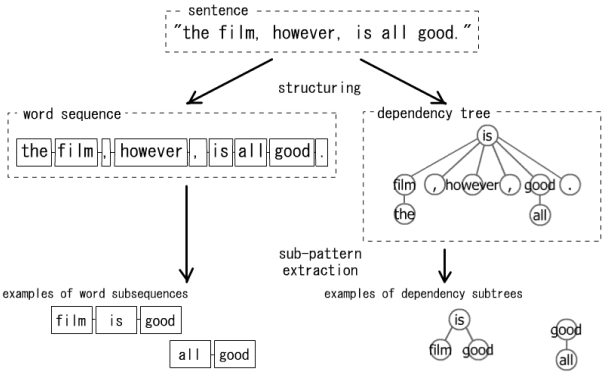

Fig. 1. A word sequence and a dependency tree representation of a sentence and examples of sub-patterns of the sentence

important and useful for sentiment classification. Thus, there appears to remain considerable room for improvement by incorporating such information.

To incorporate such information into document sentiment classification, we propose to use a word sequence and a dependency tree as structured representations of a sentence (we call simply "a sentence" below) and mining frequent sub-patterns from the sentences in a document dataset as features for document sentiment classification. We believe that the extracted set of frequent sub-patterns includes subjective expressions and idioms in the domain.

As shown in Figure 1, we regard a sentence as a word sequence and a dependency tree. We then extract frequent sub-patterns from these structured representations of sentences.

The rest of the paper is organized as follows. In the next section, we show related works on sentiment classification. In Section 3, we describe our approach to handle word order and syntactic relations between words in a sentence included in a document. In Section 4, we report and discuss the experimental results of sentiment classification. Finally, Section 5 gives conclusion.

2 Related Work

Sentiment classification is a task of classifying a target unit in a document to positive (favorable) or negative (unfavorable) class. Past researches mainly treated three kinds of target units: a word, a sentence and an overall document. to positive or negative.

Word Sentiment Classification. Hatzivassiloglou et al. [11] used conjunctive expressions such as "smart and beautiful" or "fast but inaccurate" to extract sentiment polarities of words.

Turney [7] determined the similarity between two words by counting the number of results returned by web searches. The relationship between a polarity-unknown word and a set of manually-selected seeds was used to classify the polarity-unknown word into a positive or negative class.

Sentence Sentiment Classification. Kudo et al. [9] used subtrees of word dependency trees as features for sentence-wise sentiment polarity classification. They used boosting algorithm with the subtree-based decision stamps as weak learners.

Document Sentiment Classification. Pang et al. [1] attempted to classify movie reviews. They applied to document sentiment classification a supervised machine learning method which had succeeded in other document classification tasks (e.g., on the task classifying articles of Reuters to 10 categories, Dumais et al [8] achieved F-measure of 0.92 with SVMs.). They used a *word N-gram* in the dataset as bag-of-words features for their classifier. A word N-gram is a set of N continuous words extracted from a sentence. The best results came from word unigram-based model run through SVMs, with 82.9% accuracy.

Pang [2] also attempted to improve their classifier by using only subjective sentences in the review. But accuracy of their method is less than that of the classifier using full reviews, which was introduced in their former study [1].

Dave et al. [6] used machine learning methods to classify reviews on several kinds of products. Unlike Pang's research, they obtained the best accuracy rate with word bigram-based classifier on their dataset. This result indicates that the unigram-based model does not always perform the best and that the best settings of the classifier is dependent on the data.

To use the prior knowledge besides a document, Mullen and Collier [10] attempted to use the semantic orientation of words defined by Turney [7] and several kinds of information from Internet and thesaurus. They evaluated on the same dataset used in Pang et al.'s study [1] and achieved 84.6% accuracy with the lemmatized word unigram and the semantic orientation of words.

To our knowledge, word order and syntactic relations between words in a sentence have not been used for the document sentiment classification.

3 Our Approach

We propose to use word order and syntactic relations between words in a sentence for a machine learning based document sentiment classifier. We give such information as frequent sub-patterns of sentences in a document dataset: word subsequences and dependency subtrees.

3.1 Word Subsequence

As shown in Figure 2, a word sequence is a structured representation of a sentence. From the word sequence, we can obtain ordered words in the sentence.

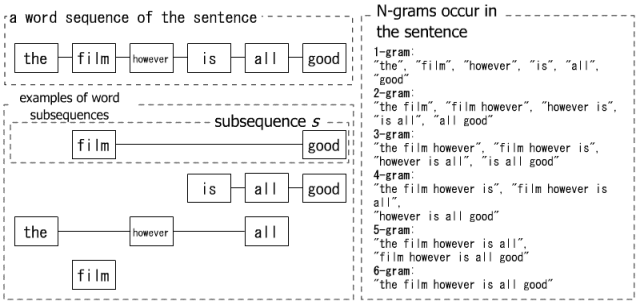

Fig. 2. A word sequence of a sentence "*The film however is all good*" and examples of subsequences

We define a *word subsequence* of a word sequence as a sequence obtained by removing zero or more words from the original sequence. In word subsequences, the word order of the original sentence is preserved.

While word N-grams cover only cooccurrences of N *continuous* words in a sentence, word subsequences cover cooccurrences of an arbitrary number of *non-continuous* words as well as continuous words. Therefore incorporating the occurrences of subsequences into the classification appears to be effective.

For example, N-grams do not cover cooccurrence of "*film*" and "*good*", when another word appears between the two words as in Figure 2. On the contrary, subsequences cover the pattern "*film-good*", denoted by s in the figure.

3.2 Dependency Subtree

As shown in Figure 3, a dependency tree is a structured representation of a sentence. The dependency tree expresses dependency between words in the sentence by child-parent relationships of nodes. We define a *dependency subtree* of a dependency tree as a tree obtained by removing zero or more nodes and branches from the original tree. The dependency subtree preserves partial dependency between the words in the original sentence. Since each node corresponding to a word is connected by a branch, a dependency subtree would give richer syntactic information than a word N-gram and a word subsequence.

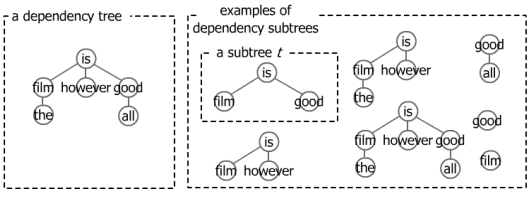

Fig. 3. A dependency tree of a sentence "*The film however is all good*" and examples of subtrees

For example, in Figure 3, to express the relation between the words "*good*" and "*film*", a dependency subtree t (denoted as *(is(film)(good))*) does not only show the cooccurrence of "*good*" and "*film*", but also guarantees that "*good*" and "*film*" are syntactically connected by the word "*is*".

3.3 Frequent Pattern Mining

The number of all sub-patterns of sentences in a document dataset tends to be very large. Thus we consider not all sub-patterns but only all *frequent* sub-patterns in the dataset. A sentence *contains* a pattern if and only if the pattern is a subsequence or a subtree of the sentence. We then define the *support* of a sub-pattern as the number of sentences containing the sub-pattern. If a support of a sub-pattern is a given *support threshold* or more, the sub-pattern is *frequent*.

We mine all frequent sub-patterns from the dataset by the following mining algorithms.

Frequent Subsequence Mining: Prefixspan [4]
Prefixspan introduced by Pei et al. [4] is an efficient algorithm for mining all the frequent subsequences from a dataset consisting of sentences. First, the algorithm starts with a set of frequent subsequences consisting of single items (in this paper, corresponding to words). Then the algorithm expands each already-obtained frequent subsequence of size k by attaching a new item to obtain frequent sequence of size $k + 1$. By repeating the latter step recursively, the algorithm obtains all frequent subsequences.

However, expanding a subsequence by attaching a new item to an arbitrary position leads to duplicated enumeration of the same candidate subsequence. To avoid such enumeration, the algorithm restricts the position to attach a new item to the end of newly-obtained subsequence in left-to-right order.

Frequent Subtree Mining: FREQT [5]
FREQT introduced by Abe et al. [5] is an efficient algorithm to mine all frequent subtrees from a dataset consisting of trees. First, the algorithm starts with a set of frequent subtrees consisting of single nodes (in this paper, corresponding to words). Then the algorithm expands each already-obtained frequent subtree of size k by attaching a new node to obtain frequent tree of size $k + 1$. By repeating the latter step recursively, the algorithm obtains all frequent subtrees.

However, expanding a subtree by attaching a new node to an arbitrary position of the subtree leads to duplicated enumeration of the same candidate subtree. To avoid such enumeration, the algorithm restricts the position to attach a new node to the end of newly-obtained subtree in depth-first order.

4 Experiment

4.1 Movie Review Dataset

We prepared two movie review datasets.

The first dataset, used by Pang et al. [1] and Mullen et al. [10], consists of 690 positive and 690 negative movie reviews. Following the experimental settings presented in Pang et al. [1] and Mullen et al. [10], we used 3-fold cross validation with this dataset for the evaluation.

The second dataset, used by Pang et al. [2], consists of 1000 positive and 1000 negative movie reviews. Following the experimental settings presented in Pang et al. [2], we used 10-fold cross validation with this dataset for the evaluation.

4.2 Features

We extract word unigram, bigram, word subsequence and dependency subtree patterns from the sentences in the dataset for features of our classifiers. Each type of features is defined as follows.

- **word unigram**: *uni*
  Unigram patterns which appear in at least 2 distinct sentences in the dataset.
- **word bigram**: *bi*
  Bigram patterns which appear in at least 2 distinct sentences in the dataset.
- **frequent word subsequence**: *seq*
  Frequent word subsequence patterns whose sizes are 2 or more. These patterns are extracted from the dataset in the fashion mentioned in Section 3.3[1]. We set the support threshold to 10. Since the number of frequent subsequences usually grows exponentially with lengths of sequences for mining, we do not use sentences but short *clauses* as the sequences. For example in Figure 4, we regard each clause of this sentence as a word sequence. We split a sentence to clauses with occurrences of the nodes labeled $'SBAR'$, which indicates a root of a subordinate clause, in a phrase structure tree of the sentence [2]. To further shorten sequences, we removed punctuations and words tagged with a part-of-speech in Table 1. Words of these parts-of-speech are presumably not a constituent of a subjective expression.

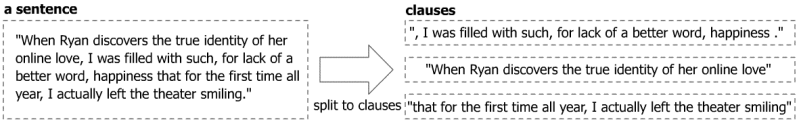

Fig. 4. An example of a sentence and clauses obtained by splitting the sentence

[1] We use Prefixspan, which is available at `http://www.chasen.org/~taku/software/`.

[2] In this paper, a phrase structure tree and part-of-speech tags of a sentence is given with Charniak parser [3].

Table 1. A list of Part-Of-Speech tags removed from word sequences of clauses

POS tag	example	POS tag	example
AUX	do done have is	NNPS	Americans Amharas
CC	and both but either	PDT	all both half many
CD	one-tenth ten million 0.5	POS	' 's
DT	all an the them these this	PRP	hers herself him himself
EX	there	PRP$	her his mine my
FW	gemeinschaft hund ich jeux	RP	aboard about across along
IN	astride among uppon whether out	SYM	% & ' "
LS	SP-44005 SP-44007 Second Third	TO	to
NNP	Motown Venneboerger Ranzer	WDT	that what whatever which

– **frequent dependency tree:** *dep*
Frequent dependency subtree patterns whose sizes are 2 or more. These patterns are extracted from the dataset in the fashion mentioned in Section 3.3[3]. We set the support threshold to 10. To avoid mining noisy patterns, we remove punctuations from the dataset.

We also extract another feature set whose elements are features consisting of lemmatized words. As in the extraction of the features *uni*, *bi*, *seq*, *dep* described above, we extract these lemmatized features (uni_l, bi_l, seq_l, dep_l).

To convert features of a document to an input of the machine-learning classifier, we define a feature vector representation of the document. Each dimension corresponds to a feature. The ith dimension's value d_i is set to 1 if the ith feature appears in this document, otherwise 0.

4.3 Classifiers and Tests

We used support vector machines (SVMs) with the linear kernel as a classifier and the feature vector representation of each document normalized by 2-norm. The linear kernel has a learning parameter C (called a soft margin parameter), which needs adjustment. Since the results of the preliminary experiments indicated the performance of the classifier are dependent on this parameter, we carried out three kinds of cross-validation tests:

– test 1: Following past researches ([1], [2], [10]), we fix C as 1 in all learning steps in each fold of the dataset. The result is used for comparison to the past researches.
– test 2: Assuming that we can select the best value for C out of $\{e^{-2.0}, e^{-1.5}, \ldots, 1, \ldots, e^{2.0}\}$, we report the average of the best accuracy rates of their folds as final result. The result shows the potential performance of the classifier with the features.

[3] We use FREQT, which is available at `http://www.chasen.org/~taku/software/`.

– test 3: In each fold, out of $\{e^{-2.0}, e^{-1.5}, \ldots, 1, \ldots, e^{2.0}\}$, we predict a proper value of C which gives the best accuracy rate of 5-fold cross-validation of the training data. We then use the predicted C for learning the training data. We report an average of these accuracy rates of their folds as final result. Since proper C cannot be obtained before experiment, the result gives the practical performance of the classifier with the features.

Since it takes too much time to evaluate all combinations of features, we first select the best combination of bag-of-words features (we denote these features as *bow*) according to the accuracy rate of test 2. We then evaluate the classifier using combinations of the *bow* and word subsequence and/or dependency subtree pattern features. We finally discuss the improvement of performance by adding the sub-pattern features with the accuracy rate of test 1 and test 3.

4.4 Result and Discussion

The results of the experiment on the dataset 1 are shown in Table 2. The results of the experiment on the dataset 2 are shown in Table 3.

Obviously, our approach is successful in both of the datasets.

In the test 1, our best classifier obtains 87.3% accuracy on the dataset 1 and 92.9% accuracy on the dataset 2. The comparison between these results and the results of past researches ([1] [10] [2]) indicates that our method is more effective for document sentiment classification than the past researches.

Table 2. Results for dataset 1

Features	Acurracy(%)		
	test1	test2	test3
Pang et al. [1]	82.9	N/A	N/A
Mullen et al. [10]	**84.6**	N/A	N/A
word unigram (= uni)	83.0	83.7	83.0
lemma unigram (= uni_l)	82.8	83.8	83.2
word bigram (= bi)	79.6	80.4	80.1
lemma bigram (= bi_l)	80.4	80.9	80.7
$uni + bi$	83.8	84.6	84.0
$uni + bi_l$	83.6	84.2	83.5
$uni_l + bi$	**84.4**	84.8	**84.6**
$uni_l + bi_l$ (= bow)	84.0	**84.9**	84.2
$bow + seq$	84.1	85.3	84.9
$bow + seq_l$	84.4	85.7	84.9
$bow + dep$	86.6	87.6	87.5
$bow + dep_l$	**87.3**	**88.3**	**88.0**
$bow + seq + dep$	86.2	87.2	87.2
$bow + seq + dep_l$	87.0	87.5	87.5
$bow + seq_l + dep$	86.5	87.5	87.0
$bow + seq_l + dep_l$	87.0	87.6	87.0

Table 3. Results for dataset 2

Features	Acurracy(%)		
	test1	test2	test3
Pang et al. [2]	**87.1**	N/A	N/A
word unigram (= uni)	87.1	88.1	87.0
lemma unigram (= uni_l)	86.4	86.9	85.9
word bigram (= bi)	84.2	85.3	85.1
lemma bigram (= bi_l)	84.3	85.2	84.7
$uni + bi$ (= bow)	**88.1**	**88.8**	**88.0**
$uni + bi_l$	87.8	88.6	87.8
$uni_l + bi$	87.3	88.2	87.3
$uni_l + bi_l$	87.7	88.3	87.9
$bow + seq$	88.2	89.4	88.3
$bow + seq_l$	88.5	89.8	88.5
$bow + dep$	92.4	**93.7**	92.7
$bow + dep_l$	92.8	**93.7**	92.9
$bow + seq + dep$	92.6	93.5	92.8
$bow + seq + dep_l$	**92.9**	**93.7**	**93.2**
$bow + seq_l + dep$	92.6	93.2	93.0
$bow + seq_l + dep_l$	**92.9**	93.3	93.1

In the test 3, our best classifier obtains 88.0% accuracy on the dataset 1 and 93.2% accuracy on the dataset 2. The comparison between these results and the results obtained by the best bag-of-words classifiers in the test 3 indicates that our classifiers are more effective for document sentiment classification than the bag-of-words based classifiers. The contribution of dependency subtree feature is large. Only using this feature with the best bag-of-words feature, we obtained obviously better performance than the preceding bag-of-words based classifiers.

Adding the word subsequence features slightly improves the baseline classifier with bag-of-words features. Since our word pruning strategies is naive, there may exist a more sophisticated strategy which gives higher performance.

The classifier using both word subsequences and dependency subtrees with the best bag-of-words features yields almost the same performance as the classifier using dependency trees and the best bag-of-words features. It suggests that there exists large overlap between these two types of pattern features.

Opposite to Pang et al. [1], using word bigrams yields good influence to the classification performance. We consider that the main reason of the difference is the setting of support threshold used to extract bigram patterns. While our method used all bigram patterns which occur at least twice in the dataset, Pang et al [1] used only patterns which occur at least 7 times in the dataset.

Lemmatized features are not always more effective for classification than the original ones. If the dataset is large, lemmatizing words may be harmful because it ignores information in the conjugated forms. If the dataset is small, subpatterns consisting of unlemmatized words tend to be infrequent. Thus there is a risk of missing sub-patterns which are useful for classification.

4.5 Weighted Patterns

A classifier, obtained by SVMs with the linear kernel, labels either of two distinct classes to examples based on a weighted voting, where each voter corresponds to a feature of SVMs. The absolute value of each weight indicates how large the contribution of the feature is.

We observed pattern features with large weights in the SVM classifier with the following features: *uni*, *bi*, *seq* and dep_l[4]. We used all reviews in the dataset 2 as training data. The value of C was set to 1. In Table 4, we show some patterns along with their weights. We could find several heavily-weighted patterns which appear to be effective to detect sentiment polarity (e.g., "*stern*", "*pull off*", "*little-life*", "*(without(doubt))*" in Table 4). We also found an overlap of patterns used for the classification. For instance, unigram pattern "*bad*" and "*movie*", bigram pattern "*bad movie*", word subsequence pattern "*bad-movie*" and dependency subtree pattern "*(movie(bad))*" overlapped each other.

[4] This combination of features follows our best classifier's settings on the dataset 2.

Table 4. Examples of patterns with their weights in the weight vector

the type of the pattern	weight	pattern
Word unigram (*uni*)	1.0618	*hilarious*
	0.6221	*masterpiece*
	-1.4265	*stern*
	-0.3749	*movie*
	-1.9937	*bad*
Word bigram (*bi*)	8.0561	*pull off*
	0.8565	*one of*
	-0.1150	*little life*
	-0.3330	*bad movie*
Word subsequence (*seq*)	-7.0200	*little-life*
	0.2340	*not-only-also*
	0.3497	*good-film*
	0.3243	*film-good*
	-0.5828	*bad-movie*
	-0.0053	*movie-bad*
Dependency subtree (dep_l)	5.9784	*(without(doubt))*
	0.0236	*(film(good))*
	-1.1406	*(should(have))*
	-0.8306	*(movie(bad))*

5 Conclusion

In this paper, we have shown the methods for incorporating word order and syntactic relations between words in a sentence into the classification. We have obtained sub-pattern features as information of word order and syntactic relations between words in a document by mining frequent sub-patterns from word sequences and dependency trees in the dataset. In the experiments on the movie review domain, our classifier with a bag-of-words feature and sub-pattern features showed better performance than past classifiers. In future work, we would like to incorporate discourse structures in a document into the classifier.

References

1. Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. Thumbs up? Sentiment Classification using Machine Learning Techniques. *Proc. of 7th EMNLP*, pp.79–86, 2002.
2. Bo Pang, Lillian Lee. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. *Proc. of 42nd ACL*, pp. 271–278, 2004.
3. Eugene Charniak. A Maximum-Entropy-Inspired Parser. *Proc. of 1st NAACL*, pp.132–139, 2000.

4. Jian Pei, Jiawei Han, Behzad Mortazavi-Asl, Helen Pinto, Qiming Chen, Umeshwar Dayal, and Mei-Chun Hsu. Prefixspan: Mining Sequential Patterns Efficiently by Prefix-Projected Pattern Growth. *Proc. of 17th ICDE*, pp.215–224, 2001.
5. Kenji Abe, Shinji Kawasoe, Tatsuya Asai, and Hiroki Arimura, Setsuo Arikawa. Optimized substructure discovery for semi-structured data. *Proc. of 6th PKDD*, pp.1–14, 2002.
6. Kushal Dave, Steve Lawrence, and David Pennock. Mining the peanut gallery: opinion extraction and semantic classification of product reviews. *Proc. of 12th WWWC*, pp.519–528, 2003.
7. Peter Turney. Thumbs up or thumbs down? semantic orientation applied to unsupervised classification of reviews. *Proc. of the 40th ACL*, pp.417–424, 2002.
8. Susan Dumais, John Platt, David Heckerman, and Mehran Sahami. Inductive Learning Algorithms and Representations for Text Categorization. *Proc. of 7th CIKM*, pp. 148–155, 1998.
9. Taku Kudo and Yuji Matsumoto. A Boosting Algorithm for Classification of Semi-Structured Text. *Proc. of 9th EMNLP*, pp.301–308, 2004.
10. Tony Mullen and Nigel Collier. Sentiment Analysis using Support Vector Machines with Diverse Information Sources. *Proc. of 9th EMNLP*, pp.412–418, 2004.
11. Vasileios Hatzivassiloglou and Kathleen McKeown. Predicting the Semantic Orientation of Adjectives. *Proc. of 35th ACL and 8th EACL*, pp. 174–181, 1997.