A Boosting-based Algorithm for Classification of Semi-Structured Text using Frequency of Substructures

Tomoya Iwakura
Fujitsu Laboratories Ltd
iwakura.tomoya@jp.fujitsu.com

Abstract

Research in text classification currently focuses on challenging tasks such as sentiment classification, modality identification, and so on. In these tasks, approaches that use a structural representation, like a tree, have shown better performance rather than a bag-of-words representation. In this paper, we propose a boosting algorithm for classifying a text that is a set of sentences represented by tree. The algorithm learns rules represented by subtrees with their frequency information. Existing boosting-based algorithms use subtrees as features without considering their frequency because the existing algorithms targeted a sentence rather than a text. In contrast, our algorithm learns how the occurrence frequency of each subtree is important for classification. Experiments on topic identification of Japanese news articles and English sentiment classification shows the effectiveness of subtree features with their frequency.

1 Introduction

Text classification is used to classify texts such as news articles, E-mails, social media posts, and so on. A number of machine learning algorithms have been applied to text classification successfully. Text classification handles not only tasks to identify topics, such as politics, finance, sports or entertainment, but also challenging tasks such as categorization of customer E-mails and reviews by types of claims, subjectivity or sentiment (Wiebe, 2000; Banea et al., 2010; Bandyopadhyay and Okumura, 2011). To identify difficult categories on challenging tasks, a traditional bag-of-words representation may not be sufficient. Therefore, a richer, structural representation is used rather than the traditional bag-of-words. A straightforward way to extend the traditional bag-of-words representation is to heuristically add new types of features such as fixed-length n-grams such as word bi-gram or tri-gram, or fixed-length syntactic relations. Instead of such approaches, learning algorithms that handle semi-structured data have become increasingly popular (Kudo and Matsumoto, 2004; Kudo et al., 2005; Ifrim et al., 2008; Okanohara and Tsujii, 2009). This is due to the fact that these algorithms can learn better substructures for each task from semi-structured texts annotated with parts-of-speech, base-phrase information or syntactic relations.

Among such learning algorithms, boosting-based algorithms have the following advantages: Boosting-based learning algorithms have been applied to Natural Language Processing problems successfully, including text classification (Kudo and Matsumoto, 2004), English syntactic chunking (Kudo et al., 2005), zero-anaphora resolution (Iida et al., 2006), and so on. Furthermore, classifiers trained with boosting-based learners have shown faster classification speed (Kudo and Matsumoto, 2004) than Support Vector Machines with a tree kernel (Collins and Duffy, 2002).

However, existing boosting-based algorithms for semi-structured data, boosting algorithms for classification (Kudo and Matsumoto, 2004) and for ranking (Kudo et al., 2005), have the following point that can be improved. The weak learners used in these algorithms learn classifiers which do not consider frequency of substructures. This is because these algorithms targeted a sentence as their input rather than a document or text consisting of two or more sentences. Therefore, even if crucial substructures appear several times in their target texts, these algorithms cannot reflect such frequency. For example, on sentiment classification, different types of negative expressions may be preferred to a positive expression which ap-

Proceedings of Recent Advances in Natural Language Processing, pages 319–326,
Hissar, Bulgaria, 7-13 September 2013.

pears several times. As a result, it may happen that a positive text using the same positive expression several times with some types of negative expressions is classified as a negative text because consideration of frequency is lacking.

This paper proposes a boosting-based algorithm for semi-structured data that considers the occurrence frequency of substructures. To simplify the problem, we first assume that a text to be classified is represented as a set of sentences represented by labeled ordered trees (Abe et al., 2002). Word sequence, base-phrase annotation, dependency tree and an XML document can be modeled as a labeled ordered tree. Experiments on topic identification of news articles and sentiment classification confirm the effectiveness of subtree features with their frequency.

2 Related Works

Prior boosting-based algorithms for semi-structured data, such as boosting algorithms for classification (Kudo and Matsumoto, 2004) and for ranking (Kudo et al., 2005), learns classifiers which do not consider frequency of substructures. Ifrim et. al (Ifrim et al., 2008) proposed a logistic regression model with variable-length N-gram features. The logistic regression learns the weights of N-gram features. Compared with these two algorithms, our algorithm learns frequency thresholds to consider occurrence frequency of each subtree.

Okanohara and Tsujii (Okanohara and Tsujii, 2009) proposed a document classification method using all substrings as features. The method uses Suffix arraies (Manber and Myers, 1990) for efficiently using all substrings. Therefore, the trees used in our method are not handled. Their method uses feature types of N-grams features, such as term frequency, inverted document frequency, and so on, in a logistic regression. In contrast, our algorithm differs in the learning of a threshold for feature values. Tree kernel (Collins and Duffy, 2002; Kashima and Koyanagi, 2002) implicitly maps the example represented in a labeled ordered tree into all subtree spaces, and Tree kernel can consider the frequency of subtrees. However, as discussed in the paper (Kudo and Matsumoto, 2004), when Tree kernel is applied to sparse data, kernel dot products between similar instances become much larger than those between different instances. As a result, this sometimes leads to overfitting in training. In contrast, our boosting algorithm considers the frequency of subtrees by learning the frequency thresholds of subtrees. Therefore, we think the problems caused by Tree kernel do not tend to take place because of the difference presented in the boosting algorithm (Kudo and Matsumoto, 2004).

3 A Boosting-based Learning Algorithm for Classifying Trees

3.1 Preliminaries

We describe the problem treated by our boosting-based learner as follows. Let $\mathcal{X}$ be all labeled ordered trees, or simply trees, and $\mathcal{Y}$ be a set of labels $\{-1, +1\}$. A labeled ordered tree is a tree where each node is associated with a label. Each node is also ordered among its siblings. Therefore, there are a first child, second child, third child, and so on (Abe et al., 2002).

Let S be a set of training samples $\{(\mathbf{x}_1, y_1), ..., (\mathbf{x}_m, y_m)\}$, where each example $\mathbf{x}_i \in \mathcal{X}$ is a set of labeled ordered trees, and $y_i \in \mathcal{Y}$ is a class label.

The goal is to induce a mapping

$$F : \mathcal{X} \rightarrow \mathcal{Y}$$

from S.

Then, we define subtrees (Abe et al., 2002).

Definition 1 *Subtree*

Let $\mathbf{u}$ *and* $\mathbf{t}$ *be labeled ordered trees. We call* $\mathbf{t}$ *a subtree of* $\mathbf{u}$*, if there exists a one-to-one mapping* φ *between nodes in* $\mathbf{t}$ *to* $\mathbf{u}$*, satisfying the conditions: (1)* φ *preserves the parent relation, (2)* φ *preserves the sibling relation, and (3)* φ *preserves the labels. We denote* $\mathbf{t}$ *as a subtree of* $\mathbf{u}$ *as*

$$\mathbf{t} \subseteq \mathbf{u} .$$

If a tree $\mathbf{t}$ is not a sbutree of $\mathbf{u}$, we denote it as

$$\mathbf{t} \nsubseteq \mathbf{u} .$$

We define the frequency of the subtree $\mathbf{t}$ in $\mathbf{u}$ as the number of times $\mathbf{t}$ occurs in $\mathbf{u}$ and denoted as

$$|\mathbf{t} \subseteq \mathbf{u}| .$$

The number of nodes in a tree $\mathbf{t}$ is referred as the size of the tree $\mathbf{t}$ and denote it as

$$|\mathbf{t}| .$$

To represent a set of labeled ordered trees, we use a single tree created by connecting the trees with the root node of the single tree in this paper. Figure 1 is an example of subtrees of a tree consisting of two sentences “a b c” and “a b” connected with the root node ®. The trees in the right box are a portion of subtrees of the left tree. Let $\mathbf{u}$ be

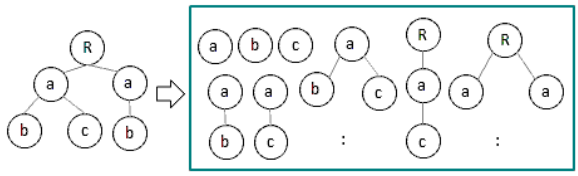

Figure 1: A labeled ordered tree and its subtrees.

the tree in the left side. For example, the size of the subtree ⓐ-ⓑ (i.e. |ⓐ-ⓑ|) is 2 and the frequency |ⓐ-ⓑ ∈ **u**| is also 2. For the subtree ⓐ-ⓒ, the size |ⓐ-ⓒ| is also 2, however, the frequency |ⓐ-ⓒ ∈ **u** | is 1.

3.2 A Classifier for Trees with the Occurrence Frequency of a Subtree

We define a classifier for trees - that is used as weak hypothesis in this paper. A boosting algorithm for classifying trees uses subtree-based decision stumps, and each decision stump learned by the boosting algorithm classifies trees whether a tree is a subtree of trees to be classified or not (Kudo and Matsumoto, 2004). To consider the frequency of a subtree, we define the following decision stump.

Definition 2 *Classifier for trees*

Let $\mathbf{t}$ *and* $\mathbf{u}$ *be trees,* z *be a positive integer, called frequency threshold, and* a *and* b *be a real number, called a confidence value, then a classifier for trees is defined as*

$$h_{\langle \mathbf{t},z,a,b\rangle}(\mathbf{u}) = \begin{cases} a & \mathbf{t} \subseteq \mathbf{u} \wedge z \leq |\mathbf{t} \subseteq \mathbf{u}| \\ -a & \mathbf{t} \subseteq \mathbf{u} \wedge |\mathbf{t} \subseteq \mathbf{u}| < z \\ b & otherwise \end{cases}.$$

Each decision stump has a subtree $\mathbf{t}$ and its frequency threshold z as a condition of classification, and two scores, a and b. If $\mathbf{t}$ is a subtree of $\mathbf{u}$ (i.e. $\mathbf{t} \subseteq \mathbf{u}$), and the frequency of the subtree $|\mathbf{t} \subseteq \mathbf{u}|$ is greater than or equal to the frequency threshold z, the score a is assigned to the tree. If $\mathbf{u}$ satisfies $\mathbf{t} \subseteq \mathbf{u}$ and $|\mathbf{t} \subseteq \mathbf{u}|$ is less than z, the score - a is assigned to the tree. If $\mathbf{t}$ is not a subtree of $\mathbf{u}$ (i.e. $\mathbf{t} \nsubseteq \mathbf{u}$), the score b is assigned to the tree.

This classifier is an extension of decision trees learned by learning algorithms like C4.5 (Quinlan, 1993) for classifying trees. For example, C4.5 learns the thresholds for features that have continuous values, and C4.5 uses the thresholds for classifying samples including continuous values. In a similar way, each decision stump for trees uses a frequency threshold for classifying samples with a frequency of a subtree.

3.3 A Boosting-based Rule Learning for Classifying Trees

To induce accurate classifiers, a boosting algorithm is applied. Boosting is a method to create a final hypothesis by repeatedly generating a weak hypothesis in each training iteration with a given weak learner. These weak hypotheses are combined as the final hypothesis. We use real AdaBoost used in BoosTexter (Schapire and Singer, 2000) since real AdaBoost-based text classifiers show better performance than other algorithms, such as discrete AdaBoost (Freund and Schapire, 1997).

Our boosting-based learner selects R types of rules for creating a final hypothesis F on several training iterations. The F is defined as

$$F(\mathbf{u}) = sign(\textstyle\sum_{r=1}^{R} h_{\langle \mathbf{t}_r,z_r a_r,b_r\rangle}(\mathbf{u})).$$

We use a learning algorithm that learns a subtree and its frequency threshold as a rule from given training samples $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{m}$ and weights over samples $\{w_{r,1}, ..., w_{r,m}\}$ as a weak learner. By training the learning algorithm R times with different weights of samples, we obtain R types of rules.

$w_{r,i}$ is the weight of sample number i after selecting $r-1$ types of rules, where $0<w_{r,i}$, $1 \leq i \leq m$ and $1 \leq r \leq R$. We set $w_{1,i}$ to $1/m$.

Let $W_{r\langle y,\leq,z\rangle}(\mathbf{t})$ be the sum of the weights of samples that satisfy $\mathbf{t} \subseteq \mathbf{x}_i$ $(1 \leq i \leq m)$, $z \leq |\mathbf{t} \subseteq \mathbf{x}_i|$ and $y_i = y$ $(y \in \{\pm 1\})$,

$$W_{r\langle y,\leq,z\rangle}(\mathbf{t}) = \sum_{i \in \{i' | \mathbf{t} \subseteq \mathbf{x}_{i'}\}} w_{r,i}[[C_{\leq}(\mathbf{x}_i, \mathbf{t}, y, z)]],$$

where $[[C_{\leq}(\mathbf{x}, \mathbf{t}, y, z)]]$ is

$$[[y_i = y \wedge z \leq |\mathbf{t} \subseteq \mathbf{x}|]]$$

and $[[\pi]]$ is 1 if a proposition π holds and 0 otherwise. Similarly, let $W_{r\langle y,<,z\rangle}(\mathbf{t})$ be the sum of the weights of samples that satisfy $\mathbf{t} \subseteq \mathbf{x}_i$, $|\mathbf{t} \subseteq \mathbf{x}_i| < z$ and $y_i = y$,

$$W_{r\langle y,<,z\rangle}(\mathbf{t}) = \sum_{i \in \{i' | \mathbf{t} \subseteq \mathbf{x}_{i'}\}} [[C_{\leq}(\mathbf{x}_i, \mathbf{t}, y, z)]],$$

where $[[C_{<}(\mathbf{x}, \mathbf{t}, y, z)]]$ is

$$[[y_i = y \wedge |\mathbf{t} \subseteq \mathbf{x}| < z]].$$

$W_{r\langle y,z\rangle}(\mathbf{t})$ is the sum of $W_{r\langle y,\leq,z\rangle}(\mathbf{t})$ and $W_{r\langle -y,<,z\rangle}(\mathbf{t})$,

$$W_{r\langle y,z\rangle}(\mathbf{t}) = W_{r\langle y,\leq,z\rangle}(\mathbf{t}) + W_{r\langle -y,<,z\rangle}(\mathbf{t}).$$

$W_{r\langle y,z\rangle}(\mathbf{t})$ means the sum of the weights of samples that are classified correctly or incorrectly with a rule, $\mathbf{t}$ and z. For example, if a confidence value of the rule is positive, $W_{r\langle +1,\leq,z\rangle}(\mathbf{t})$ is the weight

of correctly classified samples that have +1 as their labels, and $W_{r\langle -1,<,z\rangle}(\mathbf{t})$ is the weight of correctly classified samples that have -1 as their labels.

$W^{\neg}_{r\langle y\rangle}(\mathbf{t})$ is the sum of the weights of samples that a rule is not applied to (i.e. $\mathbf{t} \not\subseteq \mathbf{x}_i$) and $y_i = y$,

$$W^{\neg}_{r\langle y\rangle}(\mathbf{t}) = \sum_{i\in\{i'|\mathbf{t}\not\subseteq\mathbf{x_{i'}}\wedge y_i=y\}} w_{r,i}.$$

To select a tree $\mathbf{t}$ and a frequency threshold z the following $gain$ is used as the criterion:

$$gain(\mathbf{t}, z) \stackrel{\text{def}}{=} |\sqrt{W_{r\langle +1,z\rangle}(\mathbf{t})} - \sqrt{W_{r\langle -1,z\rangle}(\mathbf{t})}| + |\sqrt{W^{\neg}_{r\langle +1\rangle}(\mathbf{t})} - \sqrt{W^{\neg}_{r\langle -1\rangle}(\mathbf{t})}|.$$

To find the decision stump that maximizes $gain$ is the equivalent of finding the decision stump that minimizes the upper bound of the training error for real AdaBoost (Schapire and Singer, 2000; Collins and Koo, 2005). At boosting round r, a weak learner selects a subtree $\mathbf{t}_r$ ($\mathbf{t}_r \in \mathcal{X}$) and a frequency threshold z_r that maximizes $gain$ as a rule from training samples S with the weights of training samples $\{w_{r,1}, ..., w_{r,m}\}$:

$$(\mathbf{t}_r, z_r) = \mathop{\arg\max}_{(\mathbf{t}',z')\in\mathbf{ZT}} gain(\mathbf{t}', z'),$$

where $\mathbf{ZT}$ is

$$\{(\mathbf{t}, z) \mid \mathbf{t} \in \cup_{i=1}^{m}\{\mathbf{t}|\mathbf{t} \subseteq \mathbf{x}_i\} \wedge 1 \le z \le \max_{1\le i\le m} |\mathbf{t} \subseteq \mathbf{x}_i|\}.$$

Then the boosting-based learner calculates the confidence value of $\mathbf{t}_r$ and updates the weight of each sample. The confidence values a_r and b_r are defined as follows:

$$a_r = \tfrac{1}{2}\log(\tfrac{W_{r\langle +1,z\rangle}(\mathbf{t}_r)}{W_{r\langle -1,z\rangle}(\mathbf{t}_r)}), \text{ and}$$
$$b_r = \tfrac{1}{2}\log(\tfrac{W^{\neg}_{r\langle +1\rangle}(\mathbf{t}_r)}{W^{\neg}_{r\langle -1\rangle}(\mathbf{t}_r)}).$$

After the calculation of the confidence values for $\mathbf{t}_r$ and z, the learner updates the weight of each sample with

$$w_{r+1,i} = w_{r,i}\exp(-y_i h_{\langle \mathbf{t}_r, z_r a_r, b_r\rangle}(\mathbf{x}_i))/Z_r, \quad (1)$$

where Z_r is a normalization factor for $\sum_{i=1}^{m} w_{r+1,i} = 1$. Then the learner adds $\mathbf{t}_r, z_r a_r$, and b_r, to F as the r-th rule and its confidence values. The learner continues training until the algorithm obtains R rules.

3.4 Learning Rules Efficiently

We use an efficient method, rightmost-extension, to enumerate all subtrees from a given tree without duplication (Abe et al., 2002; Zaki, 2002) as

```
## S = {(x_i, y_i)}_{i=1}^m : x_i ⊆ X, y_i ∈ {±1}
## W_r = {w_{r,i}}_{i=1}^m: Weights of samples after
## learning r types of rules. w_{1,i} = 1/m
## r : The current rule number.
## The initial value of r is 1.
## T_l: A set of subtrees of size l.
## T_1 is a set of all nodes.
procedure BoostingForClassifyingTree()
While (r ≤ R)
 ## Learning a rule with the weak-learner
 {t_r, z_r} = weak-learner(T_1, S, W_r);
 ## Update weights with {t_r, z_r}
 a_r = (1/2) log( W_{r⟨+1,≤,z_r⟩}(t_r) / W_{r⟨-1,≤,z_r⟩}(t_r) )
 b_r = (1/2) log( W^¬_{r⟨+1⟩}(t_r) / W^¬_{r⟨-1⟩}(t_r) )
 ## Update weights. Z_r is a normalization
 ## factor for Σ_{i=1}^m w_{r+1,i} = 1.
 For i=1,..,m
  w_{r+1,i} = w_{r,i} exp(-y_i h_{⟨t_r, z_r a_r, b_r⟩}(x_i)) / Z_r
 r++;
end While
return F(u) = sign(Σ_{r=1}^R h_{⟨t_r, z_r a_r, b_r⟩}(u)).

## learning a rule
procedure weak-learner(T_l, S, W_r)
 ## Select the best rule from
 ## subtrees of size l in T_l.
 (t_l, z_l) = selectRule(T_l, S, W_r)
 ## If the selected (t_l, z_l) is better than
 ## current optimal rule (t_o, z_o),
 ## the (t_o, z_o) is replaced with (t_l, z_l).
 If ( gain(t_o, z_o) < gain(t_l, z_l) )
  (t_o, z_o) = (t_l, z_l);
 ## The gain of current optimal rule τ.
 τ = gain(t_o, z_o);
 ## Size constraint pruning
 If (ζ ≤ l) return (t_o, z_o);
## Generate trees that size is l + 1.
 Foreach ( t ∈ T_l)
  ## The bound of gain
  If ( u(t) < τ) continue;
  ## Generate trees of size l + 1 by rightmost
  ## extension of a tree t of size of l.
  T_{l+1} = T_{l+1} ∪ RME(t, S);
 end Foreach
 return weak-learner(T_{l+1}, S, W_r);
end procedure
```

Figure 2: A pseudo code of the training of a boosting algorithm for classifying trees.

in (Kudo and Matsumoto, 2004). The rightmost-extension starts with a set of trees consisting of single nodes, and then expands a given tree of size $k-1$ by attaching a new node to this tree to obtain trees of size k. The rightmost extension enumerates trees by restricting the position of attachment of new nodes. A new node is added to a node existing on the unique path from the root to the rightmost leaf in a tree, and the new node is added as the rightmost sibling. The details of this method can be found in the papers (Abe et al., 2002; Zaki, 2002).

In addition, the following pruning techniques are applied.

Size constraint: We examine subtrees whose size is no greater than a size threshold ζ.

A bound of gain: We use a bound of gain u(**t**):

$$u(\mathbf{t}) \stackrel{\text{def}}{=} \max_{y\in\{\pm 1\},\ 1\le z\le \max_{1\le i\le m}|\mathbf{t}\subseteq\mathbf{x}_i|} \sqrt{W_{r\langle y,z\rangle}(\mathbf{t})} + \max_{u\in\{\pm 1\}} U_{r\langle u\rangle}(\mathbf{t}),$$

where

$$U_{r\langle u\rangle}(\mathbf{t}) = |\sqrt{\textstyle\sum_{i=1}^{m} w_{r,i}[[y_i = u]]} - \sqrt{W^{\neg}_{r\langle -u\rangle}(\mathbf{t})}|.$$

For any tree $\mathbf{t}' \in \mathcal{X}$ that has $\mathbf{t}$ as a subtree (i.e. $\mathbf{t} \subseteq \mathbf{t}'$), the $gain(\mathbf{t}', z)$ for any frequency thresholds z' of $\mathbf{t}'$, is bounded under $u(\mathbf{t})$, since, for $y \in \{\pm 1\}$,

$$|\sqrt{W_{r\langle +1,z'\rangle}(\mathbf{t}')} - \sqrt{W_{r\langle -1,z'\rangle}(\mathbf{t}')}| \le \max(\sqrt{W_{r\langle +1,z'\rangle}(\mathbf{t})}, \sqrt{W_{r\langle -1,z'\rangle}(\mathbf{t})}) \le \sqrt{W_{r\langle y,z\rangle}(\mathbf{t})},$$ [1]

and

$$|\sqrt{W^{\neg}_{r\langle +1\rangle}(\mathbf{t}')} - \sqrt{W^{\neg}_{r\langle -1\rangle}(\mathbf{t}')}| \le U_{r\langle u\rangle}(\mathbf{t}),$$ [2]

where z, y and u maximize $u(\mathbf{t})$.

Thus, if $u(\mathbf{t})$ is less than or equal to the $gain$ of the current N-th optimal rule τ, candidates containing $\mathbf{t}$ are safely pruned.

Figure 2 is a pseudo code representation of our boosting-based algorithm for classifying trees. First, the algorithm sets the initial weights of samples. Then, the algorithm repeats the rule learning procedure until it obtains R rules. At each boosting round, a rule is selected by the weak-learner. The weak-learner starts to select a rule from subtrees of size 1 and the new candidates are generated by rightmost extension. After a rule is selected, the weights are updated with the rule.

4 Data Set

We used the following two data sets.

- Japanese news articles: We used Japanese news articles from the collection of news articles of Mainichi Shimbun 2010 which have at least one paragraph[3] and one of the following five categories: business, entertainment, international, sports, and technology. Table 1 shows the statistics of the Mainichi Shimbun data set. The training data is 80% of the selected news articles and test and development data are 10%. We used the text data represented by bag-of-words as well as text data represented by trees in this experiment. To convert sentences in Japanese news articles to trees, we used CaboCha (Kudo and Matsumoto, 2002), a Japanese dependency parser. [4] Parameters are decided in terms of F-measure on positive samples of the development data, and we evaluate F-measure obtained with the decided parameters. F-measure is calculated as $\frac{2\times r\times p}{p+r}$, where r and p are recall and precision.

- English Amazon review data: This is a data set from (Blitzer et al., 2007) that contains product reviews from Amazon domains. The 5 most frequent categories, book, dvd, electronics, music, and video, are used in this experiment. The goal is to classify a product review as either positive or negative. We used the file, all.review, for each domain in the data set for this evaluation. By following the paper (Blitzer et al., 2007), review texts that have ratings more than three are used as positive reviews, and review texts that have ratings less than three are used as negative reviews. We used only the text data represented by word sequences in this experiment because a parser could not parse all the text data due to either the lack of memory or the parsing speed. Even if we ran the parser for two weeks, parsing on a data set

[1] We see it from $W_{r\langle y,z'\rangle}(\mathbf{t}') \le W_{r\langle y,z'\rangle}(\mathbf{t})$ for $\mathbf{t} \subseteq \mathbf{t}'$ and $y \in \{\pm 1\}$.

[2] We see it from $W^{\neg}_{r\langle y\rangle}(\mathbf{t}) = \sum_{i\in\{i'|\mathbf{t}\not\subseteq\mathbf{x}_{i'}\wedge y_i=y\}} w_{r,i} \le \sum_{i\in\{i'|\mathbf{t}'\not\subseteq\mathbf{x}_{i'}\wedge y_i=y\}} w_{r,i} \le \sum_{1\le i\le m} w_{r,i}[[y_i = y]]$ for $\mathbf{t} \subseteq \mathbf{t}'$ and $y \in \{\pm 1\}$.

[3] There are articles that do not have body text due to copyright.

[4] http://code.google.com/p/cabocha/

Table 1: Statistics of Mainichi Shimbun data set. #P, #N and #W relate to the number of positive samples, the number of negative samples, and the number of distinct words, respectively.

Mainichi Shimbun									
Category	Training			Development			Test		
	#P	#N	#W	#P	#N	#W	#P	#N	#W
business	4,782	18,790	67,452	597	2,348	29,023	597	2,348	29,372
entertainment	938	22,632	67,682	117	2,829	29,330	117	2,829	28,939
international	4,693	18,879	67,705	586	2,359	28,534	586	2,359	29,315
sports	12,687	10,884	67,592	1,586	1,360	28,658	1,585	1,360	29,024
technology	473	23,097	67,516	59	2,887	29,337	59	2,887	28,571

Table 2: Statistics of Amazon data set. #N, #P and #W relate to the number of negative reviews, the number of positive reviews, and the number of distinct words, respectively.

Amazon review data									
Category	Training			Development			Test		
	#N	#P	#W	#N	#P	#W	#N	#P	#W
books	357,319	2,324,575	1,327,312	44,664	290,571	496,453	44,664	290,571	496,412
dvd	52,674	352,213	446,628	6,584	44,026	157,495	6,584	44,026	155,468
electronics	12,047	40,584	85,543	1,506	5,073	26,945	1,505	5,073	26,914
music	35,050	423,654	571,399	4,381	52,956	180,213	4,381	52,956	179,787
video	13,479	88,189	161,920	1,685	11,023	61,379	1,684	11,023	61,958

would not finish. Table 2 shows the statistics of the Amazon data set. Each training data is 80% of samples in all.review of each category, and test and development data are 10%. Parameters are decided in terms of F-measure on negative reviews of the development data, and we evaluate F-measure obtained with the decided parameters. The number of positive reviews in the data set is much larger than negative reviews. Therefore, we evaluated the F-measure of the negative reviews.

To represent a set of sentences represented by labeled ordered trees, we use a single tree created by connecting the sentences with the root node of the single tree.

5 Experiments

5.1 Experimental Results

To evaluate our classifier, we compare our learning algorithm with an algorithm that does not learn frequency thresholds. For experiments on Mainichi Shimbun, the following two data representations are used: Bag Of Words (BOW) (i.e. $\zeta = 1$), and trees (Tree). For the representations of texts of Amazon data set, BOW and N-gram are used. The parameters, R and ζ, are $R = 10,000$ and $\zeta = \{2, 3, 4, 5\}$.

Table 3 and Table 4 show the experimental results on the Mainichi Shimbun and on the Amazon data set. +FQ suggests the algorithms learn frequency thresholds, and -FQ suggests the algorithms do not. A McNemars paired test is employed on the labeling disagreements. If there is a statistical difference ($p < 0.01$) between a boosting (+FQ) and a boosting (-FQ) with the same feature representation, better results are asterisked (*).

The experimental results showed that classifiers that consider frequency of subtrees attained better performance. For example, Tree(+FQ) showed better accuracy than Tree(-FQ) on three categories on the Mainichi Shimbun data set. Compared with BOW(+FQ), Tree(+FQ) also showed better performance on four categories.

On the Amazon data set, N-gram(+FQ) also had better performance than BOW and N-gram(-FQ). N-gram(+FQ) performed better performances than BOW on all five categories, while performing better than N-gram(-FQ) on four categories. These results show that our proposed methods contributed to improved accuracy.

5.2 Examples of Learned Rules

By learning frequency thresholds, classifiers learned by our boosting algorithm can distinguish subtle differences of meanings. The following are some examples observed in rules learned from the book category training data. For example, three types of thresholds for "great" were learned. This seems to capture more occurrences of "great" indicated positive meaning. For classifying texts as positive, "I won't read" with $2 \leq$, which means

Table 3: Experimental Results of the training on the Mainichi Shinbun. Results in bold show the best accuracy, and while an underline means the accuracy of a boosting is better than the booting algorithm with the same feature representation (e.g. Tree(-FQ) for Tree(+FQ)) on each category.

	BOW		Tree	
Category	+FQ	-FQ	**+FQ**	-FQ
business	88.79	88.87*	**91.45***	90.89
entertaiment	95.07*	94.27	**95.11***	94.64
international	85.25	85.99*	87.91	**88.28***
sports	98.17	98.52*	**98.70***	98.64
technology	**83.02***	78.50	79.21	80.77*

Table 4: Experimental Results of the training on the Amazon data set. The meaning of results in bold and each underline are the same as Figure 3.

	BOW		N-gram	
Category	+FQ	-FQ	**+FQ**	-FQ
books	74.35*	74.13	**87.33***	87.20
dvd	83.18*	82.96	93.35	**93.66***
electronics	89.39*	89.06	93.36	**93.57***
music	77.85*	77.57	**91.65***	91.30
video	95.09*	95.04	**97.10***	96.86

more than once, was learned. Generally, "I won't read" seems to be used in negative reviews. However, reviews in training data include "I wont' read" more than once is positive reviews. In a similar way, "some useful" and "some good" with < 2, which means less than 2 times, were learned for classifying as negative. These two expression can be used in both meanings like "some good ideas in the book." or "... some good ideas, but for ... ". The learner seems to judge only one time occurrences as a clue for classifying texts as negative.

6 Conclusion

We have proposed a boosting algorithm that learns rules represented by subtrees with their frequency information. Our algorithm learns how the occurrence frequency of each subtree in texts is important for classification. Experiments with the tasks of sentiment classification and topic identification of new articles showed the effectiveness of subtree features with their frequency.

References

Kenji Abe, Shinji Kawasoe, Tatsuya Asai, Hiroki Arimura, and Setsuo Arikawa. 2002. Optimized substructure discovery for semi-structured data. In *PKDD'02*, pages 1–14.

Sivaji Bandyopadhyay and Manabu Okumura, editors. 2011. *Sentiment Analysis where AI meets Psychology*. Asian Federation of Natural Language Processing, Chiang Mai, Thailand, November.

Carmen Banea, Rada Mihalcea, and Janyce Wiebe. 2010. Multilingual subjectivity: are more languages better? In *Proc. of COLING '10*, pages 28–36.

John Blitzer, Mark Dredze, and Fernando Pereira. 2007. Biographies, bollywood, boom-boxes and blenders: Domain adaptation for sentiment classification. In *Proc. of ACL'07*, pages 440–447.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proc. of ACL'02*, pages 263–270.

Michael Collins and Terry Koo. 2005. Discriminative reranking for natural language parsing. *Computational Linguistics*, 31(1):25–70.

Yoav Freund and Robert E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences 55(1)*.

Georgiana Ifrim, Gökhan Bakir, and Gerhard Weikum. 2008. Fast logistic regression for text categorization with variable-length n-grams. In *Proc. of KDD'08*, pages 354–362.

Ryu Iida, Kentaro Inui, and Yuji Matsumoto. 2006. Exploiting syntactic patterns as clues in zero-anaphora resolution. In *Proc. of Meeting of Association for Computational Linguistics*.

Hisashi Kashima and Teruo Koyanagi. 2002. Kernels for semi-structured data. In *ICML'02*, pages 291–298.

Taku Kudo and Yuji Matsumoto. 2002. Japanese dependency analysis using cascaded chunking. In *Proc. of CoNLL'02*, pages 1–7.

Taku Kudo and Yuji Matsumoto. 2004. A boosting algorithm for classification of semi-structured text. In *Proc. of EMNLP'04*, pages 301–308, July.

Taku Kudo, Jun Suzuki, and Hideki Isozaki. 2005. Boosting-based parse reranking with subtree features. In *Proc. of ACL'05*, pages 189–196.

Udi Manber and Gene Myers. 1990. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, SODA '90, pages 319–327.

Daisuke Okanohara and Jun'ichi Tsujii. 2009. Text categorization with all substring features. In *SDM*, pages 838–846.

J. R. Quinlan. 1993. *C4.5: Programs for Machine Learning*. Morgan Kaufmann.

Robert E. Schapire and Yoram Singer. 2000. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168.

Janyce Wiebe. 2000. Learning subjective adjectives from corpora. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 735–740. AAAI Press.

Mohammed Javeed Zaki. 2002. Efficiently mining frequent trees in a forest. In *Proc. of KDD'02*, pages 71–80.