Justsystem at NTCIR-5 Patent Classification

Tetsuya TASHIRO Masaki RIKITOKU Takashi NAKAGAWA
Justsystem Corporation Aoyama Bldg.
1-2-3 Kita-Aoyama, Minato-ku, Tokyo 107-8640, Japan
{tetsuya_tashiro, masaki_rikitoku, takashi_nakagawa}@justsystem.co.jp

Abstract

Justsystem participated in Patent Classification Subtask at the Fifth NTCIR workshop. This paper overviews our machine learning-based patent application classification system. Straightforward application of Naive Bayes classifier was effective in theme categorization subtask that has a non-hierarchical category structure. In F-term categorization subtask, we regarded the complicated F-term categorization system as a tree with depth 2. We constructed the document classifier based on the Support Vector Machine and classify documents on this tree. Platt's sigmoid fitting for SVM output was used for the document ranking. We confirmed that this method was effective for this subtask.

Keywords: *Machine Learning, Text Classification, Naive Bayes, SVM*

1 Introduction

Automatic text categorization is one of the most successful applications of machine learning in the past decade. In participating in Patent Classification Subtask of NTCIR-5, we applied several machine learning techniques to categorize patent applications.

In Patent Categorization Subtask, two subtasks of 'theme categorization' and 'F-term categorization' are designed. These two subtasks have different characteristics.

In theme categorization, each target patent application has to be assigned one or more theme codes. The category structure of theme categorization is non-hierarchical. Theme categorization is regarded as a usual multiclass categorization task from its category structure. The number of categories is over 2,000 and the number of training documents is over 1.5 million. It is a relatively large scale categorization task.

In F-term categorization, given a target patent application and its theme, each application has to be assigned one or more F-terms. The category structure of F-term categorization is multi-faceted and hierarchical. A strategy for handling multi-faceted hierarchies is important in F-term categorization subtask.

We applied several statistical learning techniques to each subtask. Section 2 overviews our basic system. Section 3 describes our method applied to each subtask. Finally, we will discuss our experimental results and offer conclusions.

2 System Description

In this section, we describe an outline of our patent categorization system.

2.1 Bag-of-Words model

The bag-of-words model is probably the most widely used for modeling documents in text categorization. In this model, each document is represented as a feature vector counting the number of occurrences of different words as features and the positional and ordinal information of word occurrences is ignored. Our system employs this model.

2.2 Indexing term

Our system handles individual noun words, noun phrases (sequence of noun words) and attested subphrases as document features [2] . An attested subphrase is constituent of a longer noun phrase that also appears independently as a full noun phrase elsewhere in the document collection. The effectiveness of the phrasal indexing was examined in several past researches [3].

2.3 Feature extraction

Our system uses natural language processing methods to extract noun phrases from documents. A statical morphological analyzer are used for tokenization and part-of-speech tagging. For noun phrase identification, we apply finite state machine based grammar to the result of morphological analysis. During this process, several types of normalization are performed such as numeric normalization, normalization of long vowel markers in Katakana characters and dictionary-based normalization.

Numeric normalization

{ ‘二百万’ = 200 万’ = ‘2,000,000’ }

Long vowel marker normalization

{ ‘コンピューター’ = ‘コンピュータ’ }

Dictionary-based normalization

{ ‘プリインストール’ = ‘プレインストール’ }

After noun phrase identification is performed, the number of different feature occurrences is counted. These statistics are used in each machine learning framework.

2.4 Training of patent application classifier

A supervised learning framework is used for training of text classifier. A learning algorithm is provided a set of training documents with class label and produces a classification function that assign class labels to test documents. In formal run, we used Japanese patent applications corpus as training documents. Among the many fields of patent application, we used content fields as inputs into training part. Attribute fields such as ’name’ or ’application date’ are discarded in advance.

2.5 Patent classification

Patent application classifier receive a test application as input and performs classification with its trained classification function. The result of classification is ranked list of categories. Ranking of categories are determined according to the scores produced by each learning method.

3 Training and Classification Methods

In this section, we describe the training and classification methods used in theme categorization and F-term categorization.

3.1 Theme Categorization

3.1.1 Naive Bayes Classifier

In theme categorization subtask, we used Naive Bayes classifier with multinomial event model [7]. In the frame work of multinomial Naive Bayes classier, model parameters $P(w_t|c_j)$ and class prior probabilities $P(c_j)$ are estimated from class labeled training documents. Given these estimates and a document d_i, the probability that d_i belongs to class c_j can be determined by Bayes’ rule. Classification is performed

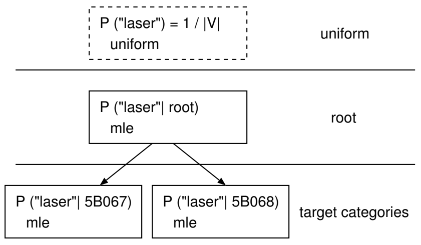

Figure 1. shrinkage-based estimation

by selecting the class with highest probability. Estimation of $P(w_t|c_j)$ and $P(c)$ is performed with word frequencies and document frequencies in class labeled training documents.

$$P(c_j|d_i) \propto P(c_j) \prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_j) \tag{1}$$

$$P(w_t|c_j) = \frac{1 + \sum_{d_i \in c_j} n(w_t, d_i)}{|V| + \sum_{t=1}^{|V|} \sum_{d_i \in c_j} n(w_t, d_i)} \tag{2}$$

$$P(c_j) = \frac{|c_j|}{\sum_{j'=1}^{|C|} |c'_j|} \tag{3}$$

$n(w_t, d_i)$ is the number of times word w_t occurs in document d_i. $|V|$ is the number of words in vocabulary.

3.1.2 Parameter estimation with shrinkage

We used statistical technique called shrinkage for improving parameter estimation [8]. Because the category structure in theme categorization is not hierarchical, we created the model with only three layers of nodes, *uniform*, *root* and target categories as illustrated in Figure 1.

In this model, parameters $P_{shrinkage}(w_t|c_j)$ are estimated by averaging three parameters in each layer with mixture weights $\lambda^0_{c_j}$, $\lambda^1_{c_j}$ and $\lambda^2_{c_j}$. At the top of the class-hierarchy, *root* class is located as a superclass of all the target categories. Smoothing is done by appending *uniform* prior beyond the root of the class hierarchy. The probability of word in *uniform* layer is given by $P_{uniform}(w_t) = 1/|V|$.

$$\begin{aligned} & P_{shrinkage}(w_t|c_j) \\ & \quad = \lambda^0_{c_j} P_{uniform}(w_t) \\ & \qquad + \lambda^1_{c_j} P_{mle}(w_t|root) \\ & \qquad + \lambda^2_{c_j} P_{mle}(w_t|c_j) \\ & s.t. \quad \sum_{k=0}^{2} \lambda^k_{c_j} = 1 \end{aligned} \tag{4}$$

Although Expectation-Maximization algorithm is used to estimate empirically optimal mixture weights in [8], we simply assigned equal value to each mixture weight.

3.1.3 Training setup

Because of the constraint of computational environment, our current system were not able to process all the training documents at once. We divided training documents into ten subsets and built ten classifiers. In dividing training documents, we made two different datasets.

TH-Dataset-1

We made TH-Dataset-1 by dividing all the training documents randomly into 10 subsets. Each subset is completely mutually exclusive.

TH-Dataset-2

Because TH-Dataset-1 contains categories with too small numbers of training document, we added extra documents to such categories from original set of training documents. We adjusted the number of training document so that each category may have at least as many as 60 documents. In this dataset, each subset is not mutually exclusive and the proportion of the training document count in each category is violated from the original dataset.

3.1.4 Classification

Test documents are classified by 10 classifiers and 10 ranked category lists are made for each test document. Final ranked list is made by integrating these 10 ranked category lists. We examined two ranking integration methods. One is *Ranking SVM* [6] and the other is *average ranking*.

In Ranking-SVM, we used *SVMLight* [4] to learn ranking function. Top 100 ranked categories from 10 lists are inputed into *SVMLight* as training data. Final ranked list is made by ranking all the categories with learned ranking model.

In average ranking, final ranking value is calculated by simply averaging 10 ranking values.

3.1.5 Evaluation

We made three runs(JSPAT1, JSPAT2, JSPAT3) varying the combination of training dataset and ranking integration method.

Table 2 shows average precision and R-precision values of the submitted results. To examine the effectiveness of ranking integration, we evaluated the accuracy of classifiers trained with 1/10 subset of the datasets. Average values of ten classifiers are also shown in Table 2.

Run ID	Integration Method	Training Dataset
JSPAT1	Ranking SVM	TH-Dataset-1
JSPAT2	Average Ranking	TH-Dataset-1
JSPAT3	Average Ranking	TH-Dataset-2

Table 1. Submitted Runs

Run ID	Avg-Precision	R-Precision
JSPAT1	0.6503	0.5507
JSPAT2	0.6591	0.5634
JSPAT3	0.6578	0.5620
Avg TH-Dataset-1	0.6419	0.5454
Avg TH-Dataset-2	0.6410	0.5444

Table 2. Result of Theme Categorization

Generally, we were not able to find significant differences between these results. The results with ranking integration methods slightly outperform the average accuracy without ranking integration. In comparing two ranking integration methods, the average ranking is better than Ranking SVM. In comparing two datasets, THEME Dataset-1 is better than THEME Dataset-2. This result probably means that adjustment of training document numbers had bad influence on class prior probability $P(c_j)$.

3.2 F-term Categorization Subtask

F-term categorization system has hierarchy structure. These hierarchy is divided into two parts: viewpoint and elements as shown by Figure:2. Each viewpoints has nested element with tree structure.

In general each theme has several hundreds of F-terms. Additionally patent documents usually have multiple F-term as shown by Table:3.

F-term categorization system are required to assign proper F-term with score from the huge categories to the each documents properly.

This subtask corresponds to the multi-category classification in the text classification area. So we used general multi-category text classifier for this subtask.

Theme	#docs	#F-term	av. #F-term
2B022	1916	5285	2.76
3G301	6699	133785	19.97
4B064	6405	54977	8.58
5H180	6222	49153	7.90
5J104	1920	19255	10.03

Table 3. Training document data for each theme

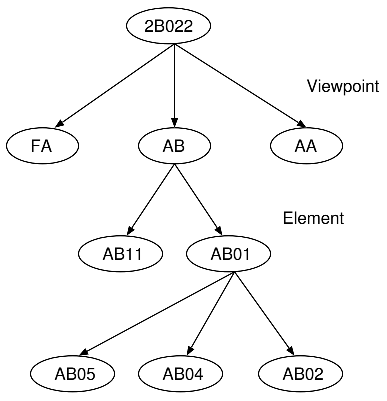

Figure 2. 2B022 F-term's viewpoint and element

RunID	Algorithm	Hierarchy
JSPAT4	sigmoid fitting SVM	2 step
JSPAT5	SVM	2 step
JSPAT6	Naive Bayes	flat F-term

Table 4. Classification Method

3.2.1 Classification Methods

In general there are several ways to classify the multi-category documents. A common one is method based on the combination of the binary classifiers. One binary classifier associated with each F-term determine whether to assign or not. A typical example of this type is one-vs-rest Support Vector Machine.

Another common one is method based on generative model. Namely, class which has posterior probability above some threshold only is assigned for the document . This type of classifier is the Naive Bayes Classifier.

In this F-term Categorization Subtask, we used three classification methods given by Table:4.

In JSPAT6 run, we used the shrinkage Naive Bayes classifier same to the Theme Categorization subtask. On this occasion, we ignored the hierarchical structure of F-term. we considered this method is baseline for this subtask.

In JSPAT4 and JSPAT5 runs, we used the method based on the Support Vector Machine(SVM) mentioned above. On this occasion, we regard the F-term categorization hierarchy as tree with depth 2:viewpoint and element, In each layer, SVM independently determine the viewpoint and element. Then, we finally determine the F-term by sum or product of outputs of each SVM.

Output score of the SVM is transformed by means of Platt's sigmoid fitting procedure [9] in JSPAT4 run. For brevity we call this method the *sigmoid fitting SVM*. and we used the output of the SVM directly in JSPAT5 run.

3.2.2 Support Vector Machine

In this subsection, we review the SVM briefly.

Let $D = \{(y_i, \mathbf{x}_i)|y_i \pm 1,\ \mathbf{x}_i \in V, i = 1 \dots N\}$ be training examples. Given example $\mathbf{x}$ corresponds to the document, SVM outputs the score by following decision function and binary value y:

$$f(\mathbf{x}) = \sum_i \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b, \qquad (5)$$

$$y(\mathbf{x}) = \begin{cases} +1 & \text{if} \quad f(\mathbf{x}) \geq 0 \\ -1 & \text{if} \quad f(\mathbf{x}) < 0 \end{cases}. \qquad (6)$$

where, $\{\alpha_i\}, b$ is determine by the following quadratic programming:

$$\text{min.} \quad \frac{1}{2}\sum_{ij}^{N} \alpha_i \alpha_j K(\mathbf{x}_i, \mathbf{x}_j) - \sum_i^N \alpha_i, \qquad (7)$$

$$\text{s.t.} \quad \sum_i^N y_i \alpha_i = 0, \qquad (8)$$
$$0 \leq \alpha_i \leq C, \quad i = 1, \dots, N.$$

where, $K(\mathbf{x}, \mathbf{y})$ is a kernel function, C is cost parameter. Please see [1], [5] for more detail of the SVM.

Values of the decision function $f(\mathbf{x})$ is depend on the problem and are not normalized. So the outputs of the different classification problem cannot be compare with each other properly. Namely we can not directly apply the ranking problem by this one-vs-rest SVM.

To overcome this problems, We normalize the outputs via Platt's sigmoid fitting method.

3.2.3 Sigmoid fitting procedure

To normalize the output of SVM and interpret as posterior probability, Platt transform the output via sigmoid function. [9]

Let $f(\mathbf{x})$ be the output of SVM for the example $\mathbf{x}$. Then, $f(\mathbf{x})$ is transformed to the posterior probability $P(y = 1|\mathbf{x})$ by following sigmoid function:

$$P(y = 1|\mathbf{x}) = \frac{1}{1 + \exp(Af(\mathbf{x}) + B)}. \qquad (9)$$

where, parameter A, B is estimated by following optimization problem:

$$\text{min.} - \sum_i \frac{y_i + 1}{2} \log p_i + \frac{1 - y_i}{2} \log(1 - p_i). \quad (10)$$

where, $p_i = P(y_i = 1|\mathbf{x}_i)$.

Note that posterior probability is normalized to $0 \leq P(y = 1|\mathbf{x}) \leq 1, P(y = 1|\mathbf{x}) + P(y = -1|\mathbf{x}) = 1$. By this transformation we can properly compare the outputs of the different problem.

F-term	JSPAT4		JSPAT5		JSPAT6	
	Av. Prec.	R-Prec.	Av. Prec.	R-Prec	Av. Prec.	R-Prec
2B022	0.370	0.326	0.299	0.235	0.458	0.390
3G301	0.356	0.366	0.212	0.252	0.299	0.288
4B064	0.427	0.452	0.212	0.225	0.371	0.362
5H180	0.508	0.469	0.257	0.253	0.411	0.384
5J104	0.338	0.328	0.123	0.134	0.320	0.299
total	0.399	0.387	0.218	0.218	0.368	0.342

Table 5. Evaluation results for each F-term

3.2.4 Training and Classify method

In this section, we give a concrete description of training and classify method of one-vs-rest SVM and their sigmoid fitting procedure for this F-term Categorization Subtask.

In training phase, the following procedure is executed.

Training:

1. Train the SVMs to classify the viewpoint by training document labeled by viewpoint.
2. For each viewpoint, train the SVMs to classify the element. In this time, we only use the training document only include the target viewpoint.
3. Estimate the sigmoid fitting parameter for sigmoid fitting SVMs to classify viewpoint and element. We use the output by cross-validation to estimate the parameter.

In classify phase, sigmoid fitting SVM classify the document as follows:

Classify:

1. Estimate the posterior probability p_1, p_2 for each viewpoint and the element via sigmoid fitting SVM.
2. Estimate the probability to assign the target F-term for each document as a $p = p_1 \times p_2$
3. If p is greater than some threshold, we assign the F-term for the target document.

In formal runs, we assumed the p_1 is uniform for all document. Namely we did not use the classifier for viewpoint.

There are two reasons for that:First, training time for the viewpoint classifier was very long and system required large memory because multi-category classification by one-vs-rest SVM was incomplete in our system. Second, classifier only for the element revealed good results in the dry run data set.

In JSPAT5 run, we used the normal SVM to measure the effect of sigmoid fitting SVM. We used linear kernel and set $C = 1$ in JSPAT4 and JSPAT5.

3.2.5 Evaluation

Table:5 shows the evaluation results for the formal run.

Sigmoid fitting SVM has performed best in 3 runs except the 2B022. It turned out that we went wrong at the 2B022 categorization procedure after the submission.

In this subtask, we only used sigmoid fitting SVM for the element and set posterior probability of the viewpoint was uniform. single element classification does not seem to be so good. Nevertheless, this method represented the best performance. We think the reason that we can properly compare the posterior probability of the different F-term via sigmoid fitting.

Shrinkage Naive Bayes classifier presented relatively good performance in all F-term Categorization Subtask. however, it revealed the lower performance than the sigmoid fitting SVM because that hierarchy structure was not considered properly.

In JSPAT5 run, we used the usual one-vs-rest SVM for element classification. Results of average precision and R.Precision together exhibited the poor performance by reason that order of output scores turns out insignificant for different elements.

In general, SVM needs more training time than other classifiers. but the SVM method used in this subtask only classify the element at once. therefore training time become relatively small because element training document set is small.

Finally, evaluation for the complete sigmoid fitting SVM for viewpoint and element is our most future work.

4 Conclusion

In this participation, we applied several machine learning techniques to categorize patent applications. Although we did not employed special techniques to address *patent-specific* features such as claim analysis or structural analysis of patent applications, our system showed good results as baseline performance in patent classification. We believe that our result is encouraging for further explorations in automated patent classification and patent analysis.

References

[1] N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[2] D. A. Evans and R. G. Lefferts. Clarit-trec experiments. In *TREC-2: Proceedings of the second conference on Text retrieval conference*, pages 385–395, Elmsford, NY, USA, 1995. Pergamon Press, Inc.

[3] S. Fujita. Notes on phrasal indexing—jscb evaluation experiments at ntcir ad hoc, 1999.

[4] T. Joachims. Making large-scale support vector machine learning practical. In A. S. B. Schölkopf, C. Burges, editor, *Advances in Kernel Methods: Support Vector Machines*. MIT Press, Cambridge, MA, 1998.

[5] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[6] T. Joachims. Optimizing search engines using clickthrough data, 2002.

[7] A. McCallum and K. Nigam. A comparison of event models for naive bayes text classification, 1998.

[8] A. K. McCallum, R. Rosenfeld, T. M. Mitchell, and A. Y. Ng. Improving text classification by shrinkage in a hierarchy of classes. In J. W. Shavlik, editor, *Proceedings of ICML-98, 15th International Conference on Machine Learning*, pages 359–367, Madison, US, 1998. Morgan Kaufmann Publishers, San Francisco, US.

[9] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. pages 61–74, 2000.