Semantic Naïve Bayes Classifier for Document Classification

How Jing, Yu Tsao
Research Center for Information Technology Innovation
Academia Sinica, Taipei, Taiwan
{yu.tsao}@citi.sinica.edu.tw

Kuan-Yu Chen, Hsin-Min Wang
Institute of Information Science
Academia Sinica, Taipei, Taiwan
{kychen, whm}@iis.sinica.edu.tw

Abstract

In this paper, we propose a semantic naïve Bayes classifier (SNBC) to improve the conventional naïve Bayes classifier (NBC) by incorporating "document-level" semantic information for document classification (DC). To capture the semantic information from each document, we develop semantic feature extraction and modeling algorithms. For semantic feature extraction, we first apply a log-Bilinear document modeling (LBDM) algorithm to transform each word into a semantic vector, and then apply principal component analysis (PCA) to the semantic space formed by the word vectors to extract a set of semantic features for each document. For semantic modeling, a semantic model is constructed using the semantic features of the training documents. In the testing phase, SNBC systematically integrates the semantic model and the conventional NBC to perform DC. The results of experiments on the 20 News-groups and WebKB datasets confirm that, with the semantic score, SNBC consistently outperforms NBC with various language modeling approaches.

1 Introduction

Document classification (DC) is an important task in the information retrieval (IR) and natural language processing (NLP) areas. In recent years, many approaches have been developed for DC. Among them, a category of approaches views DC as a traditional ranking problem. These approaches first represent a document with a feature vector, known as the vector space model (VSM), and then machine learning algorithms can be applied to accomplish classification. Notable examples belonging to this category include support vector machine (Joachims, 1998), decision tree (Comite *et al.*, 2003), logistic regression (Genkin *et al.*, 2005), and k-nearest neighbor (Kwon and Lee, 2003).

Another successful category of approaches is based on the naïve Bayes classifier (NBC). NBC assumes that a document is generated from a probabilistic model. In the offline training process, the model parameters are estimated from a set of training documents. When performing DC, NBC calculates a conditional probability $P(c|d)$ (the posterior probability that document d belongs to class c) and classifies the test document to the class that gives the highest $P(c|d)$. When calculating $P(c|d)$, NBC makes word independence (bag-of-words) assumptions and decomposes $P(c|d)$ to a product of individual word probabilities. These word probabilities are usually characterized by a statistical language model. In practice, unigram language modeling (ULM) is a popular choice due to its effectiveness and computational efficiency (Peng and Schuurmans, 2003; Bai and Nie, 2004; Wu and Wang, 2012). However, since NBC with ULM only considers frequencies of words occurring in a class, it may suffer from the problems of data sparseness and word usage diversity, leading to performance degradations for DC. To deal with the data sparseness (zero probability) problem of ULM, many smoothing techniques, such as Laplace (Bai and Nie, 2004) and Jelinek-Mercer (Jelinek and Mercer, 1980) smoothing techniques have been developed.

The latent topic language modeling (LTM) approaches use a set of latent topics to decompose the relationships between documents and classes. Successful examples include latent semantic analysis (LSA) (Bellegarda, 2005; Deerwester *et al.*, 1990), probabilistic latent semantic analysis (PLSA) (Hofmann, 1999a; Hofmann, 1999b), and latent Dirichlet allocation (LDA) (Blei 2011; Griffiths and Steyvers, 2004; Blei *et al.*, 2003). For these approaches, classification scores are not computed directly based on the frequencies of the words but instead based on a set of latent topics along with the likelihoods that each class generates the respective topics. The use of latent topics effectively tackles the word usage diversity problem for ULM and performs DC in a concept matching manner.

Although NBC with LTM approaches have taken

International Joint Conference on Natural Language Processing, pages 1117–1123,
Nagoya, Japan, 14-18 October 2013.

the semantic information into account, the document-level semantic cues are not directly incorporated for the DC task (LTM approaches only extract the word frequency information from documents). In this paper, we intend to enhance the NBC-based approaches by incorporating document-level semantic information for DC. In the training phase, we estimate the parameters of the semantic model by using the training documents. In the testing phase, a semantic score is computed based on the given test document with a particular class. The final decision of DC is made based on a combined score from the semantic model and the traditional NBC. Since the proposed framework is derived based on the NBC framework, we name it "semantic NBC" (SNBC). We conduct experiments on two sets of corpora, namely 20 Newsgroups and WebKB. Experimental results demonstrate that SNBC consistently outperforms NBC with various language modeling approaches.

The remainder of this paper is organized as follows. Section 2 defines the notations and briefly reviews the related work. Section 3 introduces the proposed SNBC framework. Section 4 describes our experimental setup and discusses experimental results. Finally, we conclude this work in Section 5.

2 Related Work

In this section, we present notation and terminology to be used in the following discussions and review related work to the proposed SNBC framework.

2.1 Notation

The basic unit in a DC task is word, which is denoted as w, where $w \in \mathbf{V}$, and $\mathbf{V}$ denotes the vocabulary set. A document is a sequence of words, and we denote a document by d, where $|d|$ represents the total number of words in the document. A class is a predefined document class, and we denote a class by c. Assuming that we have $|\mathbf{C}|$ distinct classes, the goal of DC is to classify a test document d_{test} into a specific class c, where $c \in \mathbf{C}$.

2.2 Naïve Bayes Classifier with Language Modeling

NBC performs DC in two stages: training and testing. In the training stage, a generative model is estimated based on the training documents for each class. In the testing stage, NBC calculates the posterior probability of each class given the evidence of the test document and selects the class that gives the highest probability

$$\hat{c} = \arg\max_{c} P(c \mid d). \quad (1)$$

By applying Bayes' theorem on $P(c \mid d)$, we have

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)} \propto P(d \mid c)P(c), \quad (2)$$

where $P(d \mid c)$ is the likelihood of document d under class c. Since NBC assumes that words in d are independent to each other, $P(d \mid c)$ can be decomposed as

$$P(d \mid c) = \prod_{w \in d} P(w \mid c)^{n(w,d)}, \quad (3)$$

where $P(w \mid c)$ is the class unigram probability, which indicates the likelihood of word w occurring in the class c, and $n(w,d)$ denotes the frequency of word w occurring in d. Generally, a unigram language model (ULM) is used for calculating $P(w \mid c)$. However, ULM may encounter a data sparseness (zero-probability) problem. To deal with this problem, many smoothing techniques have been developed. Laplace and Jelinek-Mercer smoothing techniques are two successful examples.

The Laplace smoothing technique calculates $P(w \mid c)$ by

$$P_{\mathrm{La}}(w \mid c) = \frac{1 + n(w,c)}{|V| + \sum_{w \in \mathbf{V}} n(w,c)}, \quad (4)$$

where $n(w,c)$ denotes the frequency of word w occurring in class c.

The Jelinek-Mercer smoothing technique calculates $P(w \mid c)$ by

$$P_{\mathrm{JM}}(w \mid c) = \lambda \frac{n(w,c)}{\sum_{w \in \mathbf{V}} n(w,c)} + (1-\lambda) P_{\mathrm{BG}}(w), \quad (5)$$

where $P_{\mathrm{BG}}(w)$ is a background language model obtained from the entire training corpus. The tunable parameter λ in Eq. (5) can be determined based on a set of development data. Although many studies have proven that NBC with ULM can provide satisfactory performance, the method has a limitation: the classification process is based on literal term matching and only considers frequencies of words occurring in a class. Thus, NBC with ULM usually suffers from the issue of word usage diversity and polysemy, which can degrade the DC performance.

2.3 Latent Topic Modeling

In contrast to literal term matching, a latent topic language modeling (LTM) incorporates a set of latent topic variables to decompose the relationships between documents and classes. PLSA (Hofmann, 1999a; Hofmann, 1999b) and LDA (Blei, 2011; Griffiths and Steyvers, 2004; Blei *et al.*, 2003) are two representative LTM approaches. For PLSA, $P(d \mid c)$ is formulated as

$$P_{\text{PLSA}}(d \mid c) = \prod_{w \in d} \left[\sum_{k=1}^{K} P(w \mid T_k) P(T_k \mid c) \right]^{n(w,d)}, \tag{6}$$

where T_k is the k^{th} latent topic variable, and K denotes the total number of latent topics. The word-topic likelihood $P(w \mid T_k)$ and topic-class likelihood $P(T_k \mid c)$ can be estimated beforehand by maximizing the total log-likelihood of the training data. For traditional NBC with ULM, the model implicitly assumes that each word in a document is drawn from a single topic distribution. On the contrary, LTM generalizes the idea to allow a mixture of latent topics, which can overcome the word diversity problem of ULM.

LDA shares a same concept as PLSA that uses a set of latent topic variables to model $P(w \mid c)$. The major difference between LDA and PLSA is that PLSA assumes the parameters of topic models to be fixed and unknown, while LDA considers the parameters as random variables that follow a Dirichlet distribution. Because LDA uses a complex form for latent topic modeling, the estimation of model parameters is hard to be solved by an exact inference. To simplify the estimation, a variety of approximation algorithms, such as the variational Bayesian expectation maximization (VBEM) (Blei, 2011; Blei *et al.*, 2003) and Gibbs sampling (Griffiths and Steyvers, 2004) algorithms, have been proposed.

2.4 Log-Bilinear Document Modeling

Log-Bilinear document modeling (LBDM) (Maas and Ng, 2010; Maas *et al.*, 2011) can be considered as a relaxed version of LTM. LBDM attempts to learn the word representation with a semantic space and use training documents to constrain those semantically similar words to be represented in near vicinity. The major difference of LBDM and LTM is that LBDM aims to directly parameterize the model for capturing word representations, while LTM focuses on estimating a set of latent topics (Maas and Ng, 2010).

For matching the empirical distribution of words in each document, LBDM introduces a document specific variable θ and defines the probability of a document as

$$\begin{aligned} P(d) &= \int P(d, \theta) d\theta \\ &= \int P(\theta) \prod_{w \in d} P(w \mid \theta)^{n(w,d)} d\theta, \end{aligned} \tag{7}$$

where

$$P(w \mid \theta) = \frac{\exp\left(\theta^T \phi_w + b_w\right)}{\sum_{w' \in \mathbf{V}} \exp\left(\theta^T \phi_{w'} + b_{w'}\right)}, \tag{8}$$

where ϕ_w is a vector representation of word w, and b_w denotes a bias for word w. LBDM assumes that ϕ_w and θ are in $\Re^{\beta}$, and the variable θ is modeled by a Gaussian prior density. The probability $P(w \mid \theta)$ indicates how close the vector representation of word w, ϕ_w, is to θ.

Assuming that we are dealing with the entire set of document collection, $\mathbf{D}$, the word representation matrix $\mathbf{R} \in \Re^{\beta \times |\mathbf{V}|}$ (the i^{th} column vector of $\mathbf{R}$ denoting the vector representation of the i^{th} word in the vocabulary) and word bias $\mathbf{b} \in \Re^{|\mathbf{V}|}$ (the i^{th} component of $\mathbf{b}$ denotes the bias term for the i^{th} word in the vocabulary) can be estimated by

$$\begin{aligned} \{\hat{\mathbf{R}}, \hat{\mathbf{b}}\} &= \arg\max_{\mathbf{R},\mathbf{b}} P(\mathbf{D}; \mathbf{R}, \mathbf{b}) \\ &= \arg\max_{\mathbf{R},\mathbf{b}} \prod_{d \in \mathbf{D}} \int P(\theta_d) \prod_{w \in d} P(w \mid \theta_d; \mathbf{R}, \mathbf{b})^{n(w,d)} d\theta_d. \end{aligned} \tag{9}$$

The integral over θ_d in Eq. (9) is intractable. To simplify the estimation, we assume that the posterior distribution is highly peaked around the MAP estimate of θ_d. By adding regularization terms for $\mathbf{R}$ and θ_d and taking the logarithm, the parameters of LBDM are approximately estimated by

$$\begin{aligned} \{\hat{\mathbf{R}}, \hat{\mathbf{b}}\} = \arg\max_{\mathbf{R},\mathbf{b}} \Bigg[& \sum_{d \in \mathbf{D}} \sum_{w \in d} n(w,d) \log P(w \mid \hat{\theta}_d; \mathbf{R}, \mathbf{b}) \\ & + \lambda \|\mathbf{R}\|^2 + \lambda \left\|\hat{\theta}_d\right\|^2 \Bigg], \end{aligned} \tag{10}$$

where $\hat{\theta}_d$ denotes the MAP estimate of θ_d for each document $d \in \mathbf{D}$. Since the objective function in Eq. (10) is not convex, a coordinate ascent process is performed to estimate the parameters. The estimation process first optimizes the word representations ($\mathbf{R}$ and $\mathbf{b}$) with keeping the MAP estimates $\hat{\theta}_d$ of each document fixed. Next, we find the new MAP estimate for each document with keeping the word representations fixed. The two estimation processes are performed iteratively until reach convergence.

When performing DC, for the class c, the MAP estimate of the variable $\hat{\theta}_c$ is estimated by using the word representations $\mathbf{R}$, $\mathbf{b}$, and a pseudo-document, which is a collection of the entire set of training documents belonging to class c. Next, the similarity between a document and a class c is determined by

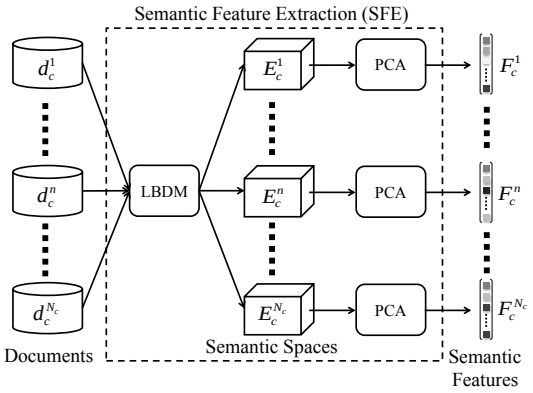

Fig. 1. Semantic feature extraction (SFE) for the c^{th} class

$$P(d \mid c) = \prod_{w \in d} P\left(w \mid \hat{\theta}_c\right)^{n(w,d)} = \prod_{w \in d} \left(\frac{\exp\left(\hat{\theta}_c^T \phi_w + b_w\right)}{\sum_{w' \in \mathbf{V}} \exp\left(\hat{\theta}_c^T \phi_{w'} + b_{w'}\right)} \right)^{n(w,d)}. \quad (11)$$

3 Semantic Naïve Bayes Classifier

NBC with LTM models has taken the semantic information into account and been confirmed effective for DC. However, the "document-level" semantic cues are not directly incorporated. In other words, documents are only treated as a sequence of words when determined the similarity between a class and a document (*cf.* Section 2). To exploit the semantic information of documents, we propose a semantic NBC (SNBC) framework in this paper. In what follows, we articulate the derivation of SNBC and the implementation process of SNBC to perform DC.

3.1 Literal and Semantic Models of SNBC

As presented in Section 2, NBC performs DC by conducting literal term matching (Eqs. (1)-(3)). To incorporate document-level semantic information, we divide $P(d \mid c)$ into two parts, namely the literal part and the semantic part, and reformulate Eq. (1) as

$$\begin{aligned} \hat{c} &= \arg\max_c P(c \mid d) \\ &= \arg\max_c P(d \mid c) P(c) \\ &= \arg\max_c P(d_s, d_l \mid c) P(c), \end{aligned} \quad (12)$$

where d_s and d_l denote the semantic and literal parts, respectively. By assuming that the literal and semantic parts are conditionally independent given a class, we have

$$P(d_s, d_l \mid c) = P(d_s \mid c) P(d_l \mid c). \quad (13)$$

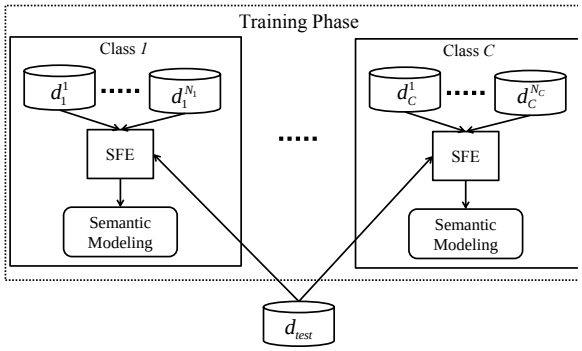

Fig. 2. Semantic modeling and semantic score calculation for a test document

As presented in Section 2, $P(d_l \mid c)$ can be estimated by NBC with any language modeling approaches. The calculation of semantic model, $P(d_s \mid c)$, will be detailed in the next section.

3.2 Document-level Semantic Information Capturing

This section describes the semantic feature extraction, semantic modeling, and semantic score calculation procedures in the SNBC framework.

3.2.1 Semantic Feature Extraction

As introduced in Section 2, LBDM can transform a word into a semantic vector representation. In the proposed SNBC framework, we incorporate LBDM to perform semantic feature extraction (SFE). Fig. 1 illustrates the SFE process. Assume that we have N_c documents in the c^{th} class. For the n^{th} document, we apply LBDM to represent each word into a semantic vector. The collection of word vectors in that document then forms a semantic subspace for that document, denoted as E_c^n. Next, we apply principal component analysis (PCA) on E_c^n to extract its principal vectors, F_c^n. Finally, we use the principal vectors F_c^n to capture main directions of semantic topics of the document d_c^n. In this paper, we only use the eigenvector with the largest eigenvalue as the semantic feature for d_c^n. We will study the use of multiple eigenvectors in our future work.

3.2.2 Semantic Modeling and Score Calculation

Figure 2 illustrates the semantic modeling process, which can be divided into training and testing stages. In the training stage, we use the semantic features of the training documents to estimate a semantic model, where each class is modeled by a semantic distribution. In this paper, we use the Gaussian mixture model (GMM) for semantic modeling. Because each class may include several sub-topics (e.g., tennis, basketball, and boxing are all categorized in the sports class), we believe that GMM is a suitable model to characterize the semantic distribution for

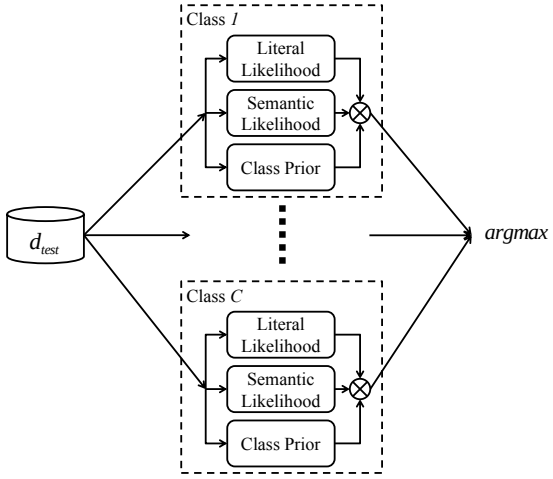

Fig. 3. SNBC score calculation for a test document

each class. The semantic features of the training documents belonging to each class are used to train the GMM for that class. In the testing stage, for each class, the class specific SFE is performed to extract the semantic feature of the test document. Next, the extracted feature is tested on the class specific GMM to obtain the semantic score, $P(d_s \mid c)$, indicating the semantic likelihood of document d on class c. Since the covariances for different GMMs may vary, we design a normalization algorithm to compensate the variations of semantic likelihoods by

$$P(d_s \mid c) = \frac{\sum_{m=1}^{M_c} \pi_m N(F_c \mid \mu_m, \Sigma_m)}{\sum_{m=1}^{M_c} \pi_m N(\mu_m \mid \mu_m, \Sigma_m)}, \quad (13)$$

where F_c is the semantic feature of the test document d_{test} for the c^{th} class, and the GMM model of class c has M_c Gaussian components with mean vectors $\{\mu_1, \ldots, \mu_{M_c}\}$, covariance matrices $\{\Sigma_1, \ldots, \Sigma_{M_c}\}$, and mixture weights $\{\pi_1, \ldots, \pi_{M_c}\}$.

3.3 SNBC Score Calculation

With NBC and the semantic model, we can calculate the final SNBC score by multiplying scores from three different information sources, namely the class prior, semantic information, and literal language modeling, as illustrated in Fig. 3. We further use α to control the scale of semantic information. Therefore, the classification rule for SNBC becomes

$$\hat{c} = \arg\max_c P(d_s \mid c)^{\alpha} P(c) \prod_{w \in d} P(w \mid c)^{n(w,d)}, \quad (14)$$

where the class prior, $P(c)$, is simply kept uniform in this paper.

4 Experiments

This section describes our experimental setup, performance measure, and experimental results.

4.1 Experimental Setup

We conducted the DC experiments on 20 Newsgroups (20Ng) and WebKB datasets (http://web.ist.utl.pt/acardoso/datasets/) (Cardoso-cachopo and Oliveira, 2003). The pre-processing steps include stemming, removing stop words, and removing numbers and words with occurrence below four. 20Ng contains roughly 20,000 documents, which distribute approximately even across 20 classes. These documents are randomly divided into 60% for training and 40% for testing. WebKB originally contains seven different categories, and we use four major classes in the experiments. Finally there are around 4,200 documents, which are randomly divided into two thirds for training and one third for testing. Albeit that the way to systemically determine the values of the parameters in various machine learning approaches is still an open issue and needs further investigation and proper experimentation, the parameters in the following experiments are set empirically as follows. The number of latent topics and the dimension of LDBM are set to 10, which gives the optimal result in our preliminary experiments. For semantic modeling, the number of GMMs equals to the number of classes, and each GMM is characterized by 20 Gaussian components. The parameter α is set to 0.6.

4.2 Performance Measure

In the following DC experiments, we use the standard F1-score measure for evaluation. F1-score (*F*) can be decomposed into two parts, namely recall (*R*) and precision (*P*).

$$R = \frac{\#\,correct\,positive prediction}{\#\,positive examples}, \quad (15)$$

$$P = \frac{\#\,correct\,positive prediction}{\#\,positive prediction}, \quad (16)$$

$$F = \frac{2 \times R \times P}{R + P}. \quad (17)$$

To evaluate the average F1-scores across all the classes, we adopt micro-averaged and macro-averaged F1-scores (Yang, 1999). The micro-averaged F1-score assigns a same weight across different classes while the macro-averaged F1-score gives each class a specific weight according to the number of documents within that class.

4.3 Experimental Results

First, we evaluate the performance of conventional NBC with various language models, including ULM with Jelinek-Mercer smoothing (JM), LDA, and LBDM; their average F1-scores evaluated on WebKB and 20Ng are listed as JM, LDA, and LBDM, respectively, in the upper three rows in Tables 1 and 2. From Tables 1 and 2, we observe that LDA outperforms JM and LBDM in most cases. The results indicate that topic modeling performs better than other language modeling approaches, which is consistent with many previous studies (Blei *et al.*, 2003).

Next, we combine the above three NBC systems, namely JM, LDA, and LBDM, with the semantic information (as presented in Section 3.3); the corresponding results are listed as SNBC (JM), SNBC (LDA), and SNBC (LBDM), respectively, in the lower three rows of Tables 1 and 2. From the experimental results, we note that SNBC consistently outperforms conventional NBC systems. The improvements from NBC to SNBC have been confirmed statistically significant based on the t-test (Agresti and Franklin, 2008). The superscript * in Tables 1 and 2 indicates that the corresponding improvement is significant at the 0.05 confidence level. Our experimental results confirm that the document-level semantic information provides complementary knowledge to the NBC with LTM and thus improve its performance for DC.

Table 1. F1-scores of NBC and SNBC on the WebKB dataset

Models	Micro-F1	Macro-F1
JM	0.8381	0.8258
LDA	0.8388	0.8253
LBDM	0.8187	0.8133
SNBC(JM)	0.8491	0.8403
SNBC(LDA)	0.8603	0.8496
*SNBC(LBDM)	0.8488	0.8346

Table 2. F1-scores of NBC and SNBC on the 20Ng dataset

Models	Micro-F1	Macro-F1
JM	0.8131	0.8070
LDA	0.8190	0.8122
LBDM	0.8144	0.8072
SNBC(JM)	0.8293	0.8152
SNBC(LDA)	0.8305	0.8213
*SNBC(LBDM)	0.8346	0.8251

5 Conclusions

This paper has proposed a semantic naïve Bayes classifier (SNBC) that incorporates document-level semantic information to improve the performance of the conventional NBC for the DC task. The SNBC framework includes a semantic feature extraction scheme to extract the semantic information of a training/test document and a semantic modeling algorithm to compute the semantic score for a given document. In the testing phase, SNBC combines the semantic score and the language modeling score to perform DC. Our experiments have been conducted on the 20 Newsgroups and WebKB datasets. The results demonstrated that SNBC can improve the DC performance in terms of Micro-F1 and Macro-F1 scores for NBC with various language modeling techniques. The performance improvement of SNBC over NBC confirms the effectiveness of integrating document-level semantic information into the conventional NBC. Notably, this study adopts LBDM to prepare semantic features; other semantic extraction methods, such as PLSA and LDA, can also be used to prepare semantic features. More experiments on SNBC using different semantic extraction methods will be conducted and compared with other existing state-of-the-art approaches in the future.

References

Alan Agresti and Christine A. Franklin. 2008. In Statistics: The Art and Science of Learning from Data. Prentice Hall.

Alexander Genkin, David D. Lewis, and David Madigan. 2005. Sparse Logistic Regression for Text Categorization. DIMACS Working Group on Monitoring Message Streams. Project Report.

Ana Cardoso-cachopo, and Arlindo L. Oliveira. 2003. An empirical comparison of text categorization methods. In String Processing and Information Retrieval, 10th International Symposium, pages 183-196.

Andrew L. Maas and Andrew Y. Ng. 2010. A probabilistic model for semantic word vectors. In Deep Learning and Unsupervised Feature Learning Workshop X NIPS.

Andrew L. Maas, Raymond E. Daly, Peter T. Pham, Dan Huang, Andrew Y. Ng, and Christopher Potts. 2011. Learning word vectors for sentiment analysis. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 142-150.

David M. Blei, Andrew Y. Ng, and Michael I. Jordan. 2003. Latent dirichlet allocation. Journal of Machine Learning Research, 3:993-1022.

David M. Blei. 2011. Introduction to probabilistic topic models. Communications of the ACM, 55(4):77–84.

Francesco De Comite, Remi Gilleron, and Marc Tommasi. 2003. Learning multi-label alternating decision trees from texts and data. In Proceedings of MLDM, pages 251-274.

Frederic Morin, and Yoshua Bengio. 2005. Hierarchical probabilistic neural network language model. In Proceedings of AISTATS, pages 246-252.

Frederick Jelinek and Robert L. Mercer. 1980. Interpolated estimation of Markov source parameters from sparse data. In Proceedings of the Workshop on Pattern Recognition in Practice, pages 381-397.

Fuchun Peng and Dale Schuurmans. 2003. Combining naïve bayes and n-Gram language models for text classification. Lecture Notes in Computer Science, 2633:335-350.

Jay M. Ponte and W. Bruce Croft. 1998. A language modeling approach to information retrieval. In the 21st International ACM SIGIR conference on research and development in Information Retrieval, pages 275-281.

Jerome R. Bellegarda. 2005. Latent semantic mapping. Signal Processing Magazine, IEEE, 22:70-80.

Jing Bai and Jian-Yun Nie. 2004. Using language models for text classification. In Proceedings of AIRS.

Meng-Sung Wu and Hsin-Ming Wang. 2012. A term association translation model for naïve bayes text classification. In Proceedings of the 16th Pacific-Asia conference on Advances in Knowledge Discovery and Data Mining - Volume Part I, pages 243-253. Springer-Verlag.

Oh-Woog Kwon and Jong-Hyeok Lee. 2003. Text categorization based on k-nearest neighbor approach for web site classification. Inf. Process. Manage., pages 25-44.

Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. Journal of the American Society for Information Science, 41:391-407.

Thomas L. Griffiths and Mark Steyvers. 2004. Finding scientific topics. In Proceedings of the National Academy of Sciences, 101:5228-5235.

Thomas Hofmann. 1999a. Probabilistic latent semantic analysis. In Proceedings of Uncertainty in Artificial Intelligence, UAI 99, pages 289-296.

Thomas Hofmann. 1999b. Probabilistic latent semantic indexing. In Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval, pages 50-57.

Thorsten Joachims. 1998. Text categorization with support vector machines: learning with many relevant features. In Proceedings of the 10th European Conference on Machine Learning, pages 137-142.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. Journal of Information Retrieval, 1:67-88.

Yoshua Bengio, Holger Schwenk, Jean-Sebastien Senecal, Frederic Morin, and Jean-Luc Gauvain. 2006. Neural probabilistic language models. In Innovations in Machine Learning. Springer Berlin/Heidelberg.