Index-based Online Text Classification for SMS Spam Filtering

Wuying Liu
School of Computer, National University of Defense Technology, Changsha, China
Email: wyliu@nudt.edu.cn

Ting Wang
School of Computer, National University of Defense Technology, Changsha, China
Email: tingwang@nudt.edu.cn

***Abstract*—We proposed a novel index-based online text classification method, investigated two index models, and compared the performances of various index granularities for English and Chinese SMS message. Based on the proposed method, six individual classifiers were implemented according to various text features of Chinese message, which were further combined to form an ensemble classifier. The experimental results from English corpus show that the relevant feature among words can increase the classification confidence and the trigram co-occurrence feature of words is an appropriate relevant feature. The experimental results from real Chinese corpus show that the performance of classifier applying word-level index model is better than the one applying document-level index model. The trigram segment outperforms the exact segment in indexing, so it is not necessary to segment Chinese text exactly when indexing by our proposed method. Applying parallel multi-thread ensemble learning, our proposed method has constant time complexity, which is critical to large scale data and online filtering.**

***Index Terms*—Online Text Classification, SMS Spam Filtering, Ensemble Learning, Index Model, Spamminess Score, TREC Spam Track**

I. INTRODUCTION

Automated Text Classification (TC), a supervised learning task of classifying natural language texts with predefined categories, has taken significant evolvements in the last 20 years [1]. Many statistical machine learning algorithms for TC have been proposed, such as Naïve Bayes (NB), k-Nearest Neighbor (kNN), Support Vector Machines (SVM), and other ensemble algorithms. Generally based on these algorithms, a classifier can be built automatically from a set of preclassified documents. TC algorithm is commonly used to resolve spam filtering problem [2] which is a special two-class classification problem and is defined as a task of classifying texts as spam or ham in an online situation.

Electronic junk, predicted about 30 years ago [3], is becoming infoglut [4] in recent years. Currently one of the most well known forms of electronic junk is mobile (Short Message Service, SMS) spam [5]. Most of previous TC algorithms apply Vector Space Model (VSM) to represent documents. But usually SMS messages are short texts, from which effective features are difficult to be extracted, and the VSM often faces the data sparse problem. Moreover, the more precise the algorithms are the more complex and time-consuming they are [6]. So, there are many new challenges for previous statistical learning algorithms in solving real-world SMS spam filtering problem which is often large-scale, high dimensional and online changeable. Hence, an effective online text classification method for classifying short messages as spam or ham has become a blooming interest.

We found that large-scale labeled messages implied the implicit classifying function which could be used to classify a new message, and then an efficient store structure was needed. Index, prevailingly employed in Information Retrieval, usually is an inverted list structure according to a hash function, so the searching operation has constant time complexity, and its incremental updating operation is also efficient. The index with advantages of searching and incremental updating can be considered as an appropriate classifying model, through comparison of the occurrence probability of an index item over different categories. So this paper adapts index to represent online classifying model and proposes an index-based online text classification method.

There are two kinds of indexes: document-level index and word-level index. In the former one, word positions within documents are not recorded, while the latter one records word positions [7]. Different level index represents different level information, so it is important to decide which level is fit for classifying model representation. A complete SMS message usually includes *FromPhoneNumber*, *ToPhoneNumber* and *BodyText* fields. The *BodyText* is a natural language text, which is fit for both document-level and word-level index models. While other structured texts, such as *FromPhoneNumber* and *ToPhoneNumber*, are only fit for document-level index model, because these "words" are position-insensitive. The feature engineering of text in one natural language is distinct from others [8].

doi:10.4304/jcp.5.6.844-851

Especially for Chinese SMS message which normally need to be segmented into tokens, we should find appropriate segment granularity of index item for *BodyText*.

To achieve the higher classifying precision in the situation of the lower computational complexity, we divide and conquer by implementing several individual classifiers and combining them to form an ensemble classifier. The individual classifier applies index-based text classification method where the index is built from one of the features of message. Ensemble learning is based on the idea that, combining individual judgments of multiple decision makers may be better than one.

This paper is organized as follows. In section II we formally define online text classification and review some most important TC algorithms. In section III, exampled as *Token* features of message body text, the index-based online text classification method is presented. In section IV, we investigate SMS spam filtering method by ensemble learning. In section V, the experiments and results are described. At last the conclusion and future work are given.

II. Related Works

Online TC is the task of assigning a *Boolean* value to each triple $<d_j, e_k, c_i> \in \mathcal{D} \times \mathcal{E} \times \mathcal{C}$, where $\mathcal{D}$ is a domain of documents, $\mathcal{E}=\{e_1, e_2, ...\}$ is a variable environment sequence, and $\mathcal{C}=\{c_1, c_2, ..., c_{|c|}\}$ is a set of predefined categories. Each environment e_k in $\mathcal{E}$ includes a set of classified documents. A value of *True* assigned to $<d_j, e_k, c_i>$ indicates a decision to document d_j under c_i in e_k environment, while a value of *False* indicates a decision not to document d_j under c_i in e_k environment. More formally, the task is to find a target function $\Phi: \mathcal{D} \times \mathcal{E} \times \mathcal{C} \rightarrow \{True, False\}$. If $\mathcal{C}$ only includes two categories {spam, ham} and each document in $\mathcal{D}$ is a SMS short message, then that is the definition of online TC for SMS spam filtering.

The NB, kNN, SVM algorithms, used to solve classical TC problem, can also be used to solve online TC problem. The NB algorithm uses the joint probabilities of words and categories to estimate the probabilities of categories given a document, whose assumption is the conditional probability of a word given a category is assumed to be independent from the conditional probabilities of other words given that category [9]. The kNN algorithm decides a document according to the k nearest neighbor document categories [10]. The online updateable NB and kNN classifier apply incremental classification model that can learn using one instance at a time. The SVM is based on the Structural Risk Minimization principle and try to find a decision surface that best separates the instances in two classes [11]. Most online SVM classifiers apply approximate methods [12]. To pursue exact global optimization, the SVM classification model can be relearn by current environment. Previous researches showed that the NB classifier could achieve higher performance, the performance of kNN and SVM were comparable, and the SVM could achieve the best performance in many tasks [6]. Except these individual TC algorithms, the ensemble algorithms are relatively new machine learning algorithms, which are based on individual classification algorithms.

Compared with the classical TC, online TC for SMS spam filtering has two important characters: one is the short text, and the other is the time-dependent environment sequence $\mathcal{E}$, which challenge previous statistical learning algorithms in effective incremental learning. The main challenges are text representation model and classification method. Previous algorithms trend to apply the VSM to represent document, and commonly there are two text representation models, one is incremental word VSM, the other is controlled word VSM. On one hand, incremental representation model has to re-represent labeled messages by current word VSM at a time of training. On the other hand, SMS spam filtering is an online open problem, whose vector space dimension also can't be fixed beforehand. Moreover, extracting effective text features is normally very difficult and the short text has to be represented as a high dimensional sparse vector. It is complex and time-consuming to update classification model frequently by above vectors. When the environment changes, the updateable NB classifier will incremental update NB classification model, and the SVM classifier can incrementally update approximately or relearn SVM classification model, while kNN needn't update its classification model. The kNN is a lazy classification method which only needs add a new classified instance into its memory. The lazy method is more suitable for SMS spam filtering because it saves time cost to update the classification model. But the kNN algorithm has a disadvantage of repeated comparison when predicting a given document, which is time-consuming. We should explore a new effective incremental learning method for online TC.

III. Index-based Online Text Classification

A. Index Model

Figure 1 shows two structures of index model that we present for online text classification. On the left, the document-level index model contains an index pair <*SpamIndex*, *HamIndex*>. In the pair, each index is an inverted list structure, which stores a list of key-value pairs <*Term*, *MessageIDset*>, where every message containing the *Term* in its body text will store its *ID* in the *MessageIDset*. On the right, the word-level index model contains two index pairs <*SpamPosIndex*, *HamPosIndex*> and <*SpamFreqIndex*, *HamFreqIndex*>. The structure of every index in the word-level model is also an inverted list structure, except that the *Key* of *PosIndex* is the combination of term string and its position in text, and the *Key* of *FreqIndex* is the combination of term string and its frequency.

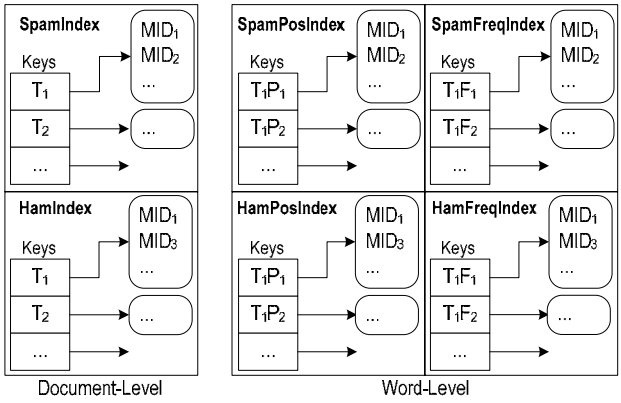

Figure 1. Index Model Structure

B. Spamminess Score

Both above two index models store two categories of index: spam and ham, which are built from labeled corpora and imply the implicit classifying function. When applying document-level index model, firstly the *BodyText* feature of a message *M* is represented by a token bag *T*. Secondly two conditional probabilities: $P((M=spam)|SpamIndex)$ and $P((M=ham)|HamIndex)$ are calculated. Finally the category of index with bigger probability is the message category. In actual application, we normalize the two conditional probabilities and use a *Spamminess Score* (*SS*) to reflect the likelihood that the processing message is spam. We assume each token has the same weight to the *SS* value, and the *SS* value of a given message is calculated as formula (1).

$$SS = \frac{N_{ham}\sum_{i=1}^{m} N_{spam}(t_i)}{N_{ham}\sum_{i=1}^{m} N_{spam}(t_i) + N_{spam}\sum_{i=1}^{m} N_{ham}(t_i)} \quad (1)$$

The symbols in formula (1) denote as follows:

- t_i -- the ith token in the token bag;
- m -- the number of tokens in the token bag;
- $N_{ham}(t_i)$ -- the number of messages whose token bag contains token t_i in *HamIndex*;
- $N_{spam}(t_i)$ -- the number of messages whose token bag contains token t_i in *SpamIndex*;
- N_{ham} -- the number of messages in *HamIndex*;
- N_{spam} -- the number of messages in *SpamIndex*.

This calculational method can be abstracted as formula (2).

$$SS \leftarrow (T, SpamIndex, HamIndex) \quad (2)$$

When applying word-level index model, firstly the *BodyText* feature is represented by two token bags: *Pos* and *Freq*. Secondly applying the calculational method like formula (2), the SS_p and SS_f are calculated separately, which are abstracted as formula (3) and formula (4).

$$SS_p \leftarrow (Pos, SpamPosIndex, HamPosIndex) \quad (3)$$

$$SS_f \leftarrow (Freq, SpamFreqIndex, HamFreqIndex) \quad (4)$$

Finally the *SS* of a message is calculated as formula (5).

$$SS=\alpha SS_p+(1-\alpha)SS_f, \text{ (where } \alpha \in [0, 1]) \quad (5)$$

In this paper, we assume position and frequency have the same weight to the *SS* value, so α=0.5.

IV. SMS Spam Filtering

A. Framework

The above section expatiated the individual classifier applying *BodyText* features (*TokenString*, *Position*, *Frequency*), and this section will combine it with other individual classifiers applying other features. Figure 2 shows our SMS spam filtering framework, which includes two parallel parts: incremental indexing and online filtering.

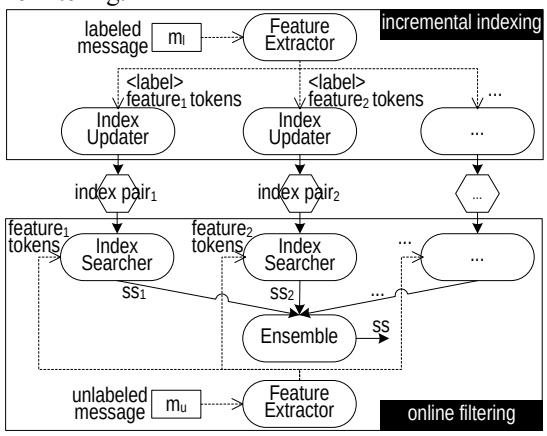

Figure 2. SMS Spam Filtering Framework

In the incremental indexing part, the *FeatureExtractor* analyzes the incoming labeled message and outputs various features of the message, and then each feature represented by tokens is used to update the corresponding index pair by its *IndexUpdater*. According to the label of message, the tokens are added into spam index or ham index.

In the online filtering part, the *FeatureExtractor* analyzes the incoming unlabeled message and outputs several features. By searching its tokens in its index pair, each *IndexSearcher* calculates its *SS*. The *Ensemble* combines several *SS* to form a final *SS* and makes a binary judgment according to a threshold of *SS*, in which there are several different ensemble functions can be chose. If each individual classifier is considered as a decision maker, then the ensemble classifier can be considered as a voting of multiple decision makers. Furthermore, applying this ensemble method the individual classifiers can parallel run.

B. Features

Except above *BodyText* features (*TokenString*, *Position*, *Frequency*), there are also several key features adapting to above index-based online text classification method, such as *Meta* features (*FromPhoneNumber, ToPhoneNumber, BodyLength*), *AcrossSentence* feature and *DistinctString* feature. The *AcrossSentence* feature reflects the inter-sentence context information, and the *DistinctString* feature mainly includes phone numbers in body text.

These features of SMS message include linguistic features of body text and behavior features of SMS message application. For instance, the linguistic features may include token string, token position, token frequency, context information, and other advanced textual semantics. While the behavior features perhaps contain

sending and receiving feature. Linguistic features have a strong discriminability in SMS spam filtering. The message language can be recognized by the character coding, and then the body text can be represented as a token list by a *Tokenizer*. Nearly all the text classifying and the text filtering problems have linguistic features. Yet the SMS spam filtering problem has its own behavior features, in which the sending and receiving feature is a more important behavior one.

After above analyses about linguistic features and behavior ones we can build several individual classifiers applying index-based online TC method easily. In order to gain better performance than the individual classifier can do, the ensemble learning based classifier makes final decision by combining multi-result of several individual classifiers.

C. Ensemble Learning

There are three advantages of the ensemble learning: statistical, computational and representational [13]. The three advantages are also reflected in the ensemble learning of SMS spam filtering. The statistical one is the ensemble multiple classification hypotheses can narrow the gap between learned classification hypotheses space and true object classification space. The computational one is that the complex object classification function can disassemble to multiple simple ones. The representational one is that the ensemble classification hypotheses can represent the new object classification hypotheses which are not included respectively by an unclose ensemble operation.

Ensemble learning mainly includes linear ensemble method and non-linear ensemble method. Applying linear ensemble method, the final *SS* is calculated as formula (6), where n means the number of individual classifiers, $\alpha_i \in [0, 1]$, $(1 \le i \le n)$ and $\sum_{i=1}^{n} \alpha_i = 1$.

$$SS = \sum_{i=1}^{n} \alpha_i SS_i \tag{6}$$

If we consider each individual classifier has the same weight to the final *SS* value, then $\alpha_i = 1/n$. We can also consider the experiential accuracy of each individual classifier as the weight α_i.

The relations of the individual classifiers are complex, not only the simple linear relations, but non-linear ones. Among non-linear ensemble methods, the SVM is fit for the non-linear sparse numerical problem. Also each individual classifier can generate a numerical *SS*. So the SVM can be used to combine multiple *SS* in non-linear space. The SVM ensemble method is showed below. Let the object space of the SVM classifier is {*spam*, *ham*}, each training vector is $<SS_1, SS_2, \ldots, SS_n>$, where SS_i denotes the *SS* by the ith individual classifier. In the learning process a SVM model can be got from the training vectors, while in the filtering process the SVM classifier can classify a new message with the SVM model, and output a final *SS*.

It is time-consuming of training a SVM model. Our previous researches show that adding a few training samples can not improve SVM performance obviously after 10,000 training samples added [14]. In order to reduce the training time and improve SVM performance obviously we compromise the variable training intervals in the immediate feedback filtering application. For instance, if there are total 80,000 messages to be filtered. Then in the first 10,000 messages, we set the training interval *zero*, that is to say there is a training process before filtering each message. In the second 10,000 messages, we set the training interval *one* that is the training process is triggered only when two new samples added. And the training interval is *three* from 20,001 to 30,000 messages, and so on. The compromise is like the delayed feedback and costs the more less training times when filtering the more last messages.

Our previous researches also show that the SVM non-linear ensemble method can get the best performance costing more time, while the arithmetic average linear ensemble method can get better performance costing less time [15]. Considering effective online SMS spam filtering application, in this paper, we assume each individual classifier has the same weight to the final *SS* value, and apply formula (6) as our ensemble function, where $\alpha_i = 1/n$.

V. Experiments

A. Corpora

Our experiments used two message corpora, one is pseudo SMS (PSMS) collection, and the other is real Chinese SMS (CSMS) collection. The PSMS collection is generated from TREC07p email set. Firstly, we extract email body text and split it to sentences. Secondly, we extract top 20 keywords from email body text. Lastly, we rank the sentences according to the number of keywords in each sentence. We regard the first rank sentence as a SMS message approximately and obtain 75,419 messages from TREC07p email set. According to the email gold standard, the PSMS collection contains 50,199 spams and 25,220 hams, and each message only includes a *BodyText* field. The CSMS collection consists of 85,870 messages, and each message includes *FromPhoneNumber*, *ToPhoneNumber* and *BodyText* fields. The CSMS collection is obtained from real volunteers and classified under two categories {spam, ham} according to their feedbacks, and then which contains 21,099 spams and 64,771 hams. Table 1 presents the detail corpora statistics.

TABLE I.
CORPORA STATISTICS

		PSMS	CSMS
quantity	spam	50,199	21,099
	ham	25,220	64,771
	total	75,419	85,870
fields	*FromPhoneNumber*		√
	ToPhoneNumber		√
	BodyText	√	√
main language		English	Chinese
source		TREC07p email set	Chinese SMS volunteers

B. Evaluation

The TREC Spam Filter Evaluation Toolkit [16] and the associated evaluation methodology are applied to evaluate classifier's performance.

There are three current static measurements:

●*hm%* -- the ham misclassification percentage;

●*sm%* -- the spam misclassification percentage;

●*lam%* -- the logistic average misclassification percentage, which is the geometric mean of the odds of *hm%* and *sm%*, converted back to a proportion.

There are two overall dynamic measurements:

●*1-ROCA%* -- the area above the ROC curve percentage;

●*h=.1%* -- one *sm%* statistic at *hm%*=0.1.

The static measurements reflect the current classifier effectiveness with the *SS* threshold, while the dynamic ones reflect the overall classifier effectiveness.

The ROC curve is the graphical representation of *hm%* and *sm%*. The area under the ROC curve is a cumulative measure of the effectiveness of the classifier over all possible values.

The ROC learning curve is the graphical representation of the classifier's behavior and the user's expectation evolve during classifier use. The ROC learning curve is that cumulative $1-ROCA\%$ is given as a function of the number of messages processed, which indicates that the classifier has reached steady-state performance.

C. Experimental Results

We apply the online immediate feedback to simulate the real online SMS spam filtering application. All the messages are processed in their real chronological sequence and the gold standard (category label) for each message is fed back to the classifier immediately after the classification decision. The running environment is *1.99GB memory* and *2.80GHz PentiumD CPU*. In the experiments, all the *SS* thresholds are set to 0.5.

The experiments consist of five parts. The Part 2 runs on the PSMS corpus, while other Parts run on the CSMS corpus. The Part 1 wants to compare the performances of previous TC methods applying full-word VSM model and our index-based TC method. Because the PSMS corpus is English text mostly, the Part 2 tries to test different index granularities in English. The Part 3 wants to compare the performances of document-level index model and word-level index model, and tries to find appropriate segment granularity of Chinese text, because the body text of Chinese short message can be represented as feature tokens in various granularities. The Part 4 investigates *BodyLength* feature and *AcrossSentence* feature. After body length statistics of CSMS messages, we try to validate the *BodyLength* feature is useful. After testing different *AcrossSentence* features, we try to find appropriate *AcrossSentence* feature token. The Part 5 compares the performances of six individual classifiers and the ensemble one.

Part 1

Applying proposed index-based TC method, we implemented a classifier in *BodyText* field named ITC. We also compare three previous classifiers (NB, kNN, SVM) in the same field. The SVM classifier comes from LIBSVM [17], while other two come from WEKA 3-6-0 [18]. The NB is updateable NB classifier; the kNN is updateable IBK classifier.

There are total five runs in this part. Applying full-word VSM, the NB, kNN1 (means k=1), kNN45 (means k=45), SVM classifiers run once separately. The kNN45 had the better performance in previous classical TC research [6]. At last the base dimensions of full-word VSM reached 100,618. Applying word-level index model, our ITC runs once with a word token segmented by a Chinese wordlist. Table II presents the processing time, the current and overall measurements. Which shows our index-based TC method precedes others, because the ITC classifier achieves the best *h=.1%* value (9.51), and the second best *1-ROCA%* value (0.1773) in the lowest processing time (11 seconds).

TABLE II.
EXPERIMENTAL RESULTS STATISTICS (PART 1)

	current (*threshold*=0.5)			overall		processing time
	hm%	sm%	lam%	1-ROCA%	h=.1%	(seconds)
NB	0.60	1.45	0.94	0.9428	82.80	1,108
kNN1	20.98	1.05	5.05	5.0791	98.82	126,027
kNN45	8.65	10.78	9.66	3.4628	71.79	126,843
SVM	2.39	3.31	2.82	0.5127	41.53	18,754
ITC	2.43	0.90	1.48	0.1773	9.51	11

Part 2

Applying word-level index model the ITC runs with three tokens: *Unigram* run whose token is the single word, *Bigram* run whose token is the sequence of 2 words, and *Trigram* run whose token is the sequence of 3 words. Figure 3 shows the Part 2 ROC curves, where *Bi*, *Tri*, *Uni* mean *Bigram*, *Trigram*, *Unigram* runs respectively. From which, we find the *Trigram* run achieves the best performance. Because trigram co-occurrence token is a very important classification feature, which implies more relevant feature among words.

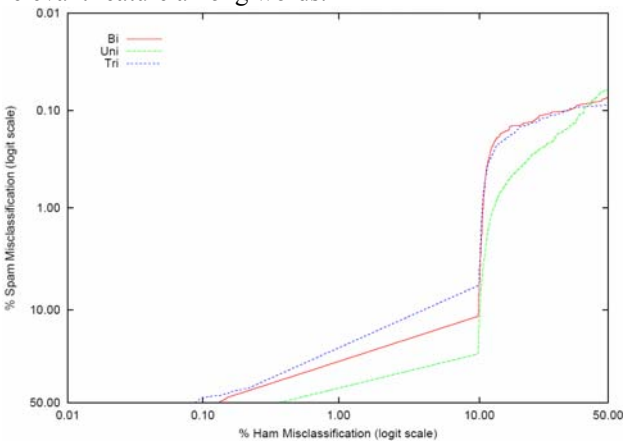

Figure 3. ROC Curves (Part 2)

Figure 4 illustrates three 2-dimension *SS* spaces according to above three runs separately. The horizontal coordinates indicate SS_p and the vertical coordinates indicate SS_f. Each SMS message is represented as a point (SS_p, SS_f). Figure 4 also validates the relevant feature among words is a key classification feature, and the trigram co-occurrence feature can increase the classification confidence of spam and ham.

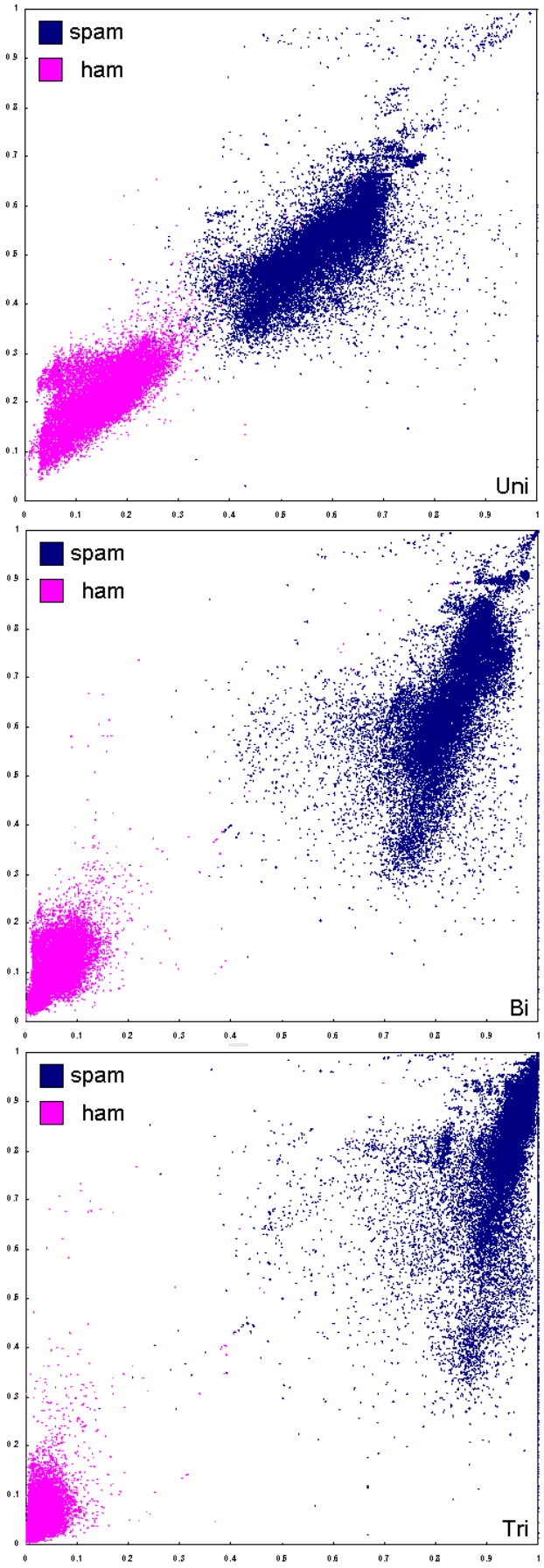

Figure 4. 2-dimension *SS* Spaces

Part 3

Our ITC runs with four tokens: *Unigram* whose token is the single Chinese character, *Bigram* whose token is the sequence of 2 Chinese characters, *Trigram* whose token is the sequence of 3 Chinese characters and *Word* whose token is a word segmented by a Chinese wordlist. Figure 5 shows the Part 3 ROC curves, where *Bi*, *Tri*, *Uni*, *Word* mean *Bigram*, *Trigram*, *Unigram*, *Word* runs respectively, and *Index1*, *Index2* mean document-level index model and word-level index model respectively. Comparing the curves of *TriIndex1*, *TriIndex2* in the Figure 5, we find the performance of word-level index model is better than document-level index model. Comparing the curves of *BiIndex1*, *TriIndex1*, *UniIndex1* and *WordIndex1* in the Figure 5, we find the trigram run achieves the best performance. It is also the same result in the *Index2* situation. Because trigram co-occurrence token implies more relevant feature among Chinese characters, it suggests that exact segment is not needed in indexing Chinese text by our proposed method.

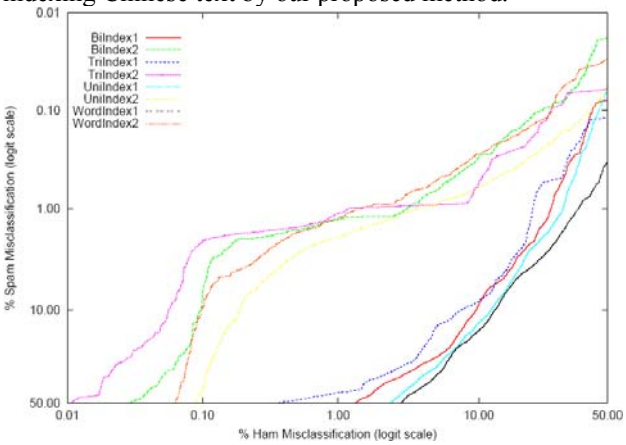

Figure 5. ROC Curves (Part 3)

We also run our ITC with four tokens applying *Index2* in one thread as well as two threads and record their time costs respectively. Figure 6 shows that the time cost of *Index2* (two threads) is almost half of *Index2* (one thread), and closes to the *Index1* time. So the parallel running can reduce time cost and has constant time complexity.

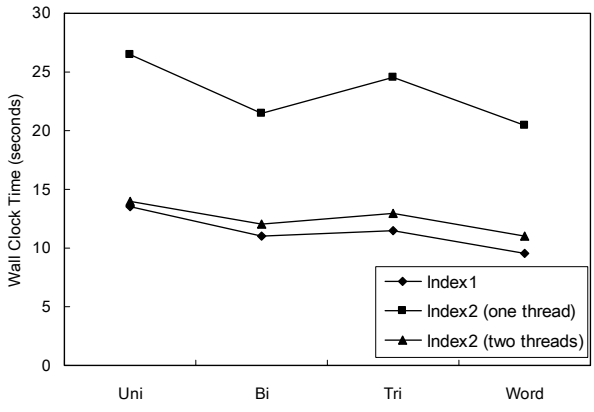

Figure 6. Time Costs (Part 3)

Part 4

Figure 7 shows the body length distribution of spam and ham is obvious distinctive in CSMS collection. This distribution is consistent with reality, because the SMS message is interactive information, the body length of most SMS hams are shorter. But the body length of SMS spams, in order to convey more complete information, must be longer relatively. From Figure 7 we can see that the length of most ham message is less than 50 characters. This length distribution rule can be used as an important feature. In this paper, we use the number of body characters as an index token.

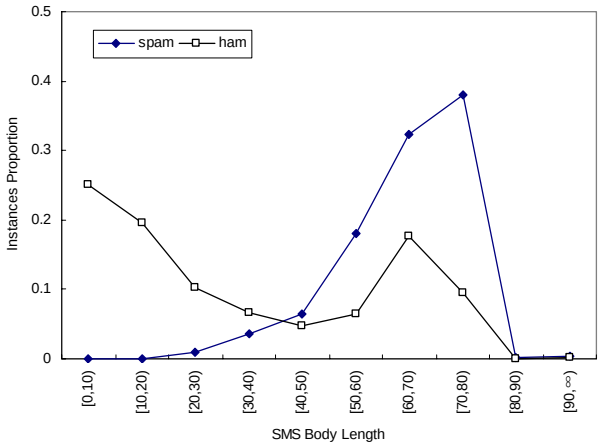

Figure 7. CSMS Body Length Statistics (Part 4)

We connect *n*-character before punctuation, punctuation, and *n*-character after punctuation to form a token to represent the *AcrossSentence* feature. We test *n* from 1 to 10, and the *1-ROCA%* performance is showed in the Figure 8. We find the Pun2 (*n=2*) feature achieves the best performance.

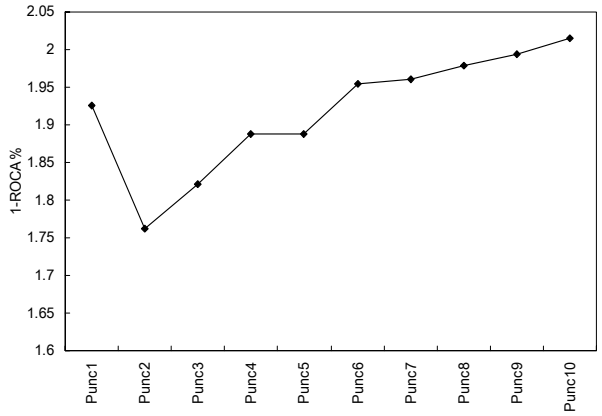

Figure 8. 1-ROCA% Performance (Part 4)

Part 5

We combined six individual classifiers to form an ensemble classifier. Table III presents the experimental results statistics, which shows that the *Ensemble* classifier achieves the best dynamic measurements, though some of its static measurements, such as *sm%*, are not the best. It is because the *SS* threshold preset at 0.5 is not optimized for the classifiers. Note that spam filtering pursues to reach the lower *lam%* in the situation of the lower *hm%*, and the lower *hm%* is a prerequisite for practical using of the method. In this sense, the *Ensemble* classifier outperforms others with the lowest *hm%* (0.03) by costing some *sm%*, which led to an advantage on the lowest *1-ROCA%* (0.0032) and the lowest *h=.1%* (0.05). To sum up, the overall performance of the *Ensemble* classifier is the best among them.

TABLE III.
EXPERIMENTAL RESULTS STATISTICS (PART 5)

	current (*threshold*=0.5)			overall		processing time
	hm%	sm%	lam%	1-ROCA%	h=.1%	(seconds)
FromPhoneNumber	3.80	0.01	0.19	1.4596	73.65	1
ToPhoneNumber	0.36	6.23	1.53	0.2555	21.84	2
BodyPhoneNumber	98.38	0.00	0.00	10.3280	20.82	3
BodyLength	14.62	3.07	6.86	4.6097	93.79	1
Punc2	61.25	0.11	4.07	1.7617	6.34	7
WordIndex2	2.43	0.90	1.48	0.1773	9.51	11
Ensemble	0.03	0.13	0.06	0.0032	0.05	29

We can also evaluate the work from ROC analysis. Figure 9 shows the Part 5 ROC curves and Figure 10 shows the Part 5 ROC learning curves. In addition, the learning speed of the *Ensemble* classifier is faster than others. Especially in the horizontal coordinates range from 0 to 30000 in the Figure 10, our *Ensemble* classifier reaches steady-state more quickly.

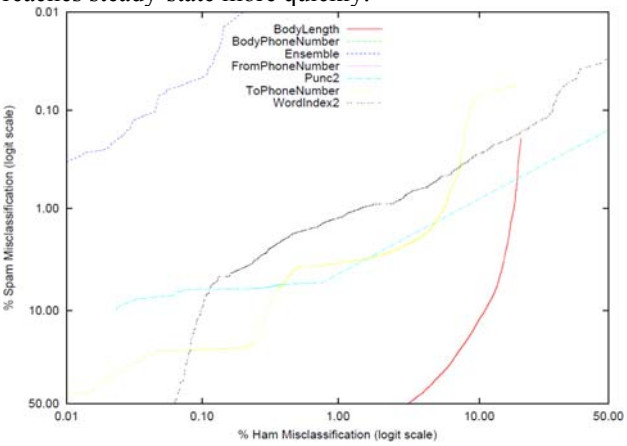

Figure 9. ROC Curves (Part 5)

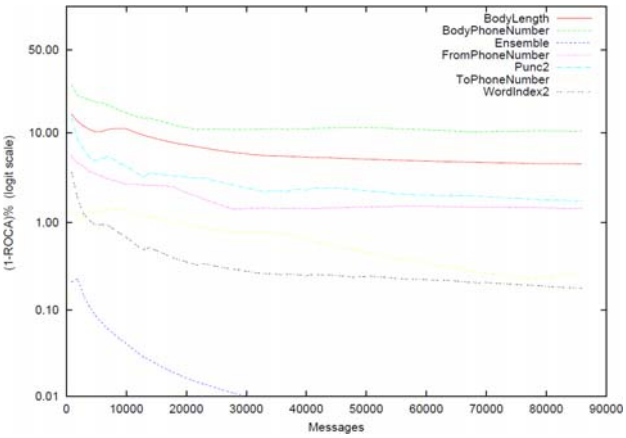

Figure 10. ROC Learning Curves (Part 5)

VI. CONCLUSION

In this paper, two index-based classifying models are proposed, based on which a novel online text classification method is presented. In our experiments of SMS spam online filtering, the ensemble classifier combining six individual classifiers by several key features achieves the best filtering effect. This ensemble method reduces time cost efficiently from three parallel levels. The first level is between incremental indexing and online filtering, the second is among individual classifiers, and the third is between indexes of each index pair. This research reveals that the trigram token index is more effective for Chinese SMS text, compared to other index granularities. This observation gives a suggestion that in the task of indexing Chinese SMS text by our proposed method, exact segment is not so important as in other NLP tasks.

This research also finds that the finer index achieves higher precision. So, future research will concern semantic features of short message by adding semantic role information to index. We will also add some important computable behavioral features, such as social network, to refine the index model for better classification effect.

ACKNOWLEDGMENT

This research is supported by the National Natural Science Foundation of China (No. 60873097) and the Program for New Century Excellent Talents in University (No. NCET-06-0926). This research is also funded by the Excellent Ph.D. Candidate Grants of NUDT (No. B080605).

REFERENCES

[1] Fabrizio Sebastiani, "Machine Learning in Automated Text Categorization," *ACM Computing Surveys*, vol. 34, no. 1, March 2002, pp.1–47.
[2] José María Gómez Hidalgo, Guillermo Cajigas Bringas, Enrique Puertas Sánz, "Content Based SMS Spam Filtering," DocEng'06, Amsterdam, The Netherlands, 2006.
[3] Denning, P. J, "Electronic Junk," *ACM Communications*, vol. 25, no. 3, March 1981, pp.163–165.
[4] Denning, P. J, "Infoglut," *CACM IT Profession*, July 2006.
[5] G. V. Cormack, José María Gómez Hidalgo, Enrique Puertas Sánz, "Spam Filtering for Short Messages," CIKM'07, Lisboa, Portugal, 2007.
[6] Yang Y. M. and Liu X, "A Re-Examination of Text Categorization Methods," SIGIR99, 1999, pp. 42–49.
[7] Justin Zobel and Alistair Moffat, "Inverted files for text search engines," *ACM Computing Surveys*, vol. 38, no. 2, Article 6, 2006.
[8] G. V. Cormack, José María Gómez Hidalgo, Enrique Puertas Sánz, "Feature Engineering for Mobile (SMS) Spam Filtering," SIGIR'07, Amsterdam, The Netherlands, 2007.
[9] George H. John, Pat Langley. Estimating continuous distributions in bayesian classifiers. In Proc. Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 1995, pp.338–345.
[10] D. Aha, D. Kibler. Instance-based learning algorithms. Machine Learning, 1991, 6: 37–66.
[11] Thorsten Joachims. Text categorization with support vector machines: learning with many relevant features. In Proc. European Conference on Machine Learning (ECML), Berlin, 1998, pp.137–142.
[12] M. Martin. On-line support vector machines for function approximation. Techn. report, Universitat Politècnica de Catalunya, Departament de Llengatges i Sistemes Informàtics, 2002.
[13] Thomas G. Dietterich, "Ensemble methods in machine learning," Multiple Classifier Systems (MCS2000), Cagliari, Italy, 2000.
[14] Wuying Liu, Ting Wang, "Active Learning for Online Spam Filtering," *Information Retrieval Technology: Proceedings of the 4th Asia Information Retrieval Symposium*, Springer, ISBN 978-3-540-68633-0, 2008.
[15] Wuying Liu, Ting Wang, "Online Spam Filtering Based on Ensemble Learning of Multi-filter," *Journal of Chinese Information Processing*, vol.22, no.1, pp.67–73, 2008.
[16] G. V. Cormack and T. R. Lynam, "TREC 2005 Spam Track Overview," TREC 2005 Proceedings, 2005.
[17] Chih-Chung Chang, Chih-Jen Lin, "LIBSVM: a Library for Support Vector Machines," February 27, 2009.
[18] Ian H. Witten and Eibe Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2nd Edition, Morgan Kaufmann, San Francisco, 2005.

Wuying Liu was born in 1980. He received his B.S. degree and Master degree in computer science and technology from National University of Defense Technology in 2002 and 2005 respectively. Currently, he is a Ph.D. candidate at School of Computer, National University of Defense Technology. His research interests include Text Classification, Information Filtering and Machine Learning.

Ting Wang was born in 1970. He received his B.S. degree and Ph.D. degree in computer science and technology from National University of Defense Technology in 1992 and 1997 respectively. Currently, he is a professor and doctoral supervisor at National University of Defense Technology. His research interests include Natural Language Processing and Computer Software. He has published more than 40 papers and undertaken two NSFC projects and two 863 projects as principal investigator.