SCAI Experiments on TREC-9

Yu-Hwan Kim, Sun Kim, Jae-Hong Eom, and Byoung-Tak Zhang

Artificial Intelligence Lab (SCAI)
School of Computer Science and Engineering
Seoul National University
Seoul 151-742, Korea
{yhkim, skim, jheom, btzhang}@scai.snu.ac.kr
http://scai.snu.ac.kr/

Abstract

In TREC-9 our main concerns cover Filtering, Web, and QA. As before, we focused on applying machine learning techniques to TREC tasks. We tackled the Filtering track by employing a boosted naive Bayes (BayesBoost) algorithm. The Web track introduced HTML tag importance factors and optimized them by genetic algorithms. Finally, our QA system is based on a natural language processing approach. Our experiments show that these algorithms are promising and competitive to existing techniques.

1 Introduction

In TREC-9, the Artificial Intelligence Lab at SNU (SCAI) participated in 3 different tracks: Filtering, Web, and QA. All of them are our second entry to the TREC. For filtering experiments, we used Boosting techniques which are popular in the machine learning community. Boosting aims to make better classifiers by combining marginally efficient classifiers. Especially, we used the boosted naive Bayes (BayesBoost) algorithm, which is shown to be a promising technique in text filtering [7]. Unfortunately, our results showed only moderate performance which is in contrast to our past experiments. We assume this is partially due to our poor feature selection methods. In addition to the old BayesBoost algorithm, our new algorithm contains some improvements including cost-sensitive learning and document-length normalization. For Web-document retrieval, our experiments have focused on the optimization of HTML tag importance factors using genetic algorithms. Additionally, we tried another weighting methods such as a 2-poisson model and an inference networks model, which improved the overall performance of the system. For QA, we used a traditional natural language based approach which results in a moderate improvement of the overall performance. We also introduced an answer-text-segment decision method based on polygon center points.

The paper is organized as follows. In Section 2, we describe and report on the Filtering experiments, which is followed by the results on the Web experiments in Section 4 and QA experiments in Section 5. Section 6 summarizes our experiments at TREC-9.

2 Filtering Track

We used a boosted naive Bayes algorithm, which we call BayesBoost for short. Boosting is a method which aims to produce a ultimate classifier by combining several marginally efficient classifiers. Especially, AdaBoost is gaining popularity in the machine learning community. Recently, AdaBoost is applied to TREC filtering and showed good performance [4]. In AdaBoost, each weak learner (i.e., a classifier performing just slightly better than random guessing) iteratively learns documents re-weighted by the previous weak learner. The combined classifier is composed by weighted voting of weak learners.

In the original AdaBoost it is restricted that each hypothesis or weak learner can produce an output -1 or 1. A more sophisticated version of the AdaBoost algorithm is AdaBoost with confidence ratio [3]. It interprets the sign of the weak learner , say $\theta_t(x)$, as the predicted label (-1 or 1) to be assigned to instance, say x, and the magnitude $|\theta_t(x)|$ as the "confidence" in this prediction. Thus if $\theta_t(x)$ is close to or far from zero, it is interpreted as a low or high confidence prediction. In this paper, the range of θ_t is [-1,+1].

AdaBoost builds a classifier by combining the hypotheses:

$$f(x) = \sum_{t=1}^{T} \alpha_t \theta_t(x),$$

where T is the number of iterations and α_t is the voting weight for each hypothesis θ_t. Several weak learners can be used in boosting. In our previous paper, we used a modified naive Bayes algorithm which enables AdaBoost to calculate confidence ratios in an easy and natural way. It also reported that Boosted naive Bayes showed a promising performance when tested on the TREC-7 and TREC-8 datasets. In naive Bayes classification the probability of a document being relevant (or irrlevant) can be formulated as follows:

$$P(d_i|c_i = j; \theta) = P(|d_i|) \prod_{k=1}^{|d_i|} P(w_{d_{ik}}|c_i = j; \theta)^{N(w_{d_{ik}}, d_i)}. \tag{1}$$

After given the probability for each class (relevance or irrelevance), naive Bayes classifiers only have to choose the most probable class. Here, a traditional naive Bayes classifier assumes that $P(|d_i|)$ is uniform, which is far from ideal. Instead, we assumed that $P(|d_i|)$ follow Gaussian distribution. This can be justified by the following argument. In contrast to other data, only a few terms for a document are available. Because of this variability of documents the

Table 1: The performance of BayesBoost on TREC-9 Filtering datasets.

	submitted run		bug fixed run	
Topics	OHSUMED	SAMP	OHSUMED	SAMP
Avg. T9U	2.78	20.0	3.75	22.14

classification of short documents can be seriously misleading. For instance, assume that we have a document whose content is only 'the'. Though this document is meaningless most of the cases, if, unfortunately, the probability 'the' occurs in a relevant document is bigger than the probability of 'the' occurs in a irrelevant document this document will be classified as relevant, which is not true. As the cost of retrieving false-positives is bigger than false-negatives, we should suppress the probability of short documents to be retrieved. Additionally in many cases, the average length of the positive documents is much longer than that of the relevant documents, which might be due to the fact that longer documents have more information and therefore more probable to be relevant. Thus, when given short documents using a Gaussian distribution instead of a uniform has the effect of suppressing short documents to be retrieved. If the length of a document is long enough we assume that this distribution is uniform as before.

We also needed to modify the original AdaBoost in order to maximize the linear utility measure. Original AdaBoost tries to minimize the classification error. However, in Filtering track we should maximize the linear utility, $T9U$. By changing the objective function of AdaBoost and therefore changing some equations used in AdaBoost we could significantly improve the performance.

We submitted our runs for OHSUMED and MESH-SAMP topics, where we used only $T9U$ for a relevance judgement measure. The submitted runs are summerized in Table 2. One thing to note is that our system had a minor bug when we submitted our runs, resulting in degraded performance. Our bug fixed version showed in 3.75 average T9U which is around 1.0 point better than our submitted runs which reported only 2.78 average $T9U$ in the OHSUMED dataset. As we had little time for parameter fitting, we just used parameters fitted for the TREC-7 AP documents set. For preprocessing of the documents, we extracted about 5000 terms in the increasing order of document frequency. After removing terms in the stop-list, we performed a simple tf-idf weighting. We used neither MESH headings nor other information except the .W and .T fields. Though relevant documents could be classified 'definitely relevant' and 'possibly relevant', we did not used this information on training the BayesBoost classifiers. We did not even used topic descriptions. Most of the time, no more than 20 rounds of Boosting steps, the training accuracy went up to 1. We stopped training when all the training documents were perfectly classified.

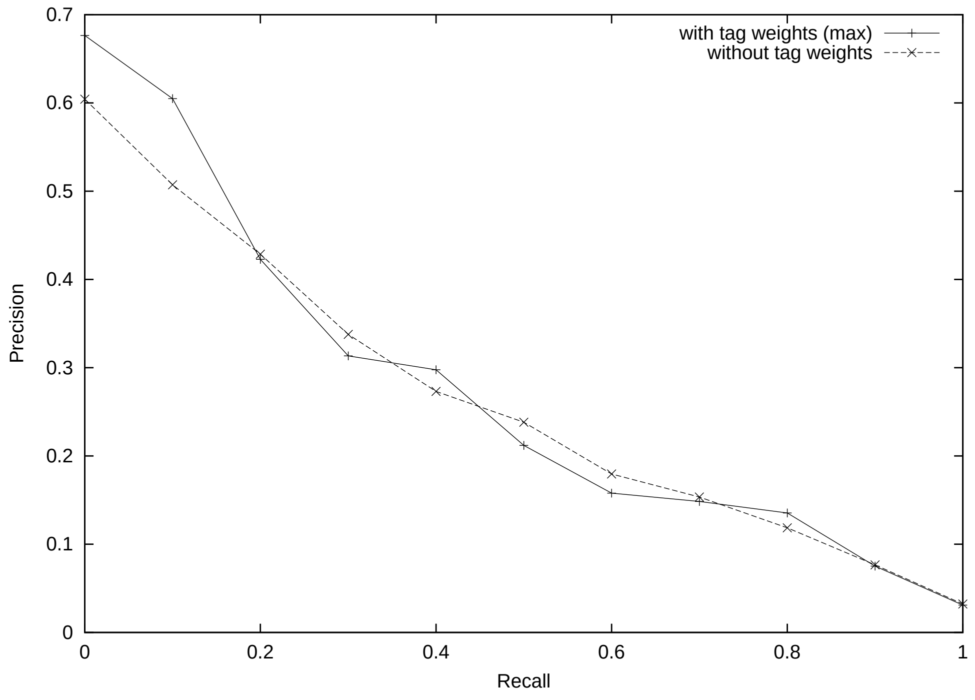

Figure 1: Interpolated recall-precision averages for Web track

3 Web Track

Web-document retrieval is based on the SCAIR (SCAI information retrieval engine) [5]. Our concern centered around the following two questions. (1) Is it useful to use HTML structures for improving Web-document retrieval performance, and (2) How can the average precision be changed by weighting schemes.

We introduced a new approach that uses HTML structures for improving retrieval performance [6]. The method was tested on WT2g (TREC-8 document set). Unfortunately, the method was not tested to WT10g yet. Our approach is based on the idea that HTML tags contain information for determining the significance of documents. We first selected a set of tags that were considered to be significant. Next, the importance factors for the tags were learned using a genetic algorithm. The modified cosine similarity was used for comparing the tag importance factors:

$$sim(d_i, q_j) = \frac{\sum_{k=1}^{n} \alpha_{ik} \cdot w_{ik} \cdot q_{jk}}{\sqrt{\sum_{k=1}^{n} (\alpha_{ik} \cdot w_{ik})^2 \cdot \sum_{k=1}^{n} q_{jk}^2}}, \qquad (2)$$

where w_{ik} is the weight of the kth term in the ith document, q_{jk} is the weight of the kth term in the jth query, and for all the tags which are determined by

Weighting scheme	11-point average precision
tf.idf	0.0972
Inference network	0.1424
2-poisson	0.1224

Table 2: Average precision for weighting schemes on Web track data

term k, and α_{ik} is the product of the tag weights.

For experiments, five tags (<TITLE>, <H>, <B>, <I>, and <A>) were selected. 10 queries (Topics 401 to 410) were used for learning the tag importance factors and another 10 queries (Topics 411 to 420) were used for retrieval. Figure 1 shows the precision-recall curves for using and not using the tag weights. As the result, we found the tag importance factors can help to get higher precision at low recall.

To answer the second question, three weighting schemes were experimented on TREC-9 document set. It was performed to know the reason why we did not get satisfactory results. The base weighting scheme is tf.idf, which is used for SCAIR. An inference network and a 2-poisson model are applied for the comparison of the retrieval performances using tf.idf [1][2]. For experiments, the parameters of the retrieval engine were optimized for tf.idf schemes using the TREC-8 data set. Only the weighting equations were replaced. The results are shown to Tables 2 and 2. For the baseline method, the inference network, and the 2-poisson model, the average precisions are 0.0972, 0.1424, and 0.1224, respectively. The result shows that the performance using two other schemes (inference network and 2-possion) are better than using tf.idf even though the parameter optimization is not used. But, still there is room for future improvement because 0.1424 is not a good average precision. There are problems to be solved such as noise correction and feature selection.

4 Question Answering

Our question answering system is simply constructed based on the TREC-8 adhoc engine [5]. Our system consists of three major components, i.e. an adhoc document retrieval component, a passage retrieval component, and an answering-zone selection component.

For document retrieval, we modified the ScaiTREC-8 adhoc engine slightly for the QA task. Using this component, we indexed all the documents producing inverted files. We also considered word normalization at document term indexing step, which is not considered in the TREC-8 QA track. As can be seen in Table 3, this is an improvement to the result of TREC-8 QA 250-bytes. The query expansion and any tagging was not considered in our TREC-9 QA track. For each question, the top 50 documents are considered as relevant documents.

For passage retrieval, we considered the distribution of query terms and term

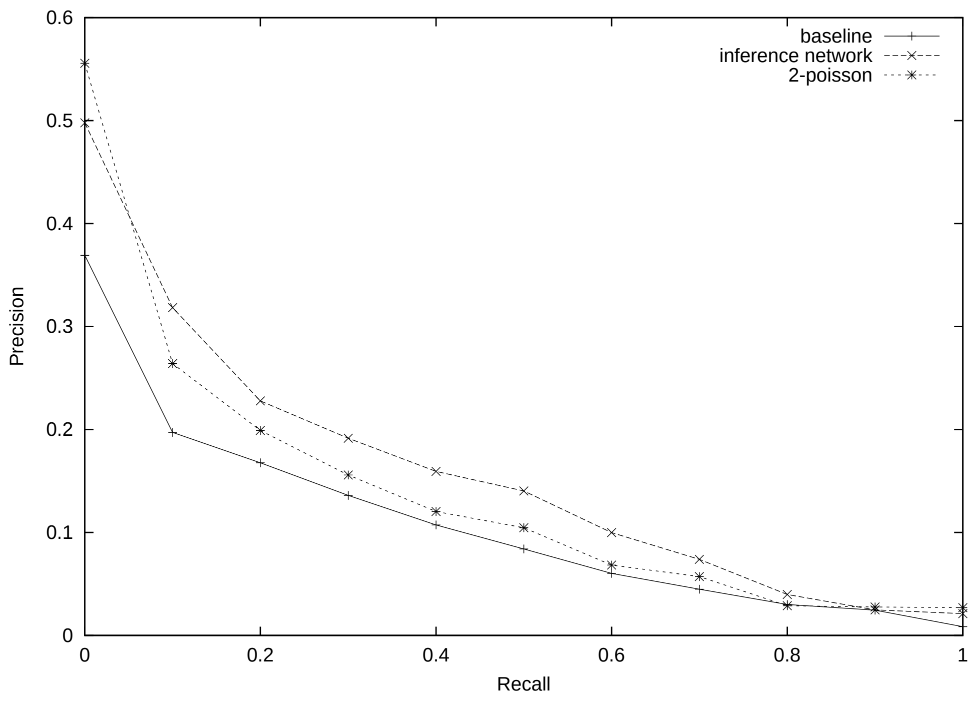

Figure 2: Comparison of recall-precision averages for weighting schemes on Web track

weights over the retrieved documents to select candidate passages in passage retrieval components. Let D be the retrieved documents and i be the passage index in D. The score of passage i, $SCORE_{D(i)}$ was calculated as follows:

$$SCORE_{D(i)} = \sum_{j=1}^{N}(\alpha_j I_j), \tag{3}$$

where N is the total number of terms, α_j is a weight of the j-th query term and I_j is an indicator variable which gets value 1 if this term appears in passage i, and gets value 0 in other cases.

Then, we selected 5 passages over the passage candidate in the decreasing order of scores. Selected passages are passed to the answering-zone selection module to extract answer text segments with proper length. Sentence boundaries are recognized by a tokenizer, which was used in E-TRAN98 English to Korean translation package [8], to extract non-stale passage structure. A word token which ends in a separator character is regarded as a definition of sentence boundaries. The tokenizing routine is applied to each of the top-ranked docu-

	50-bytes run	250-bytes run
Mean reciprocal rank (strict)	0.101	0.217
Mean reciprocal rank (lenient)	0.111	0.232

Table 3: TREC-9 QA results

ments to divide it into "sentences." We also considered all the other exceptional cases with this package.

To find a position of an answer string in the extracted passage from the retrieved relevant documents, we considered the center point of the polygon generated by term occurrence position. The center point of polygon was adjusted by each term's weight, and then text segments were extracted. For the 50-byte category, 22 bytes before the center point and 25 bytes after the center point were extracted. For the 250-byte category, 102 bytes before the center point and 147 bytes after the center point were extracted.

We submitted the results obtained by the two co-occurrence scoring methods: one in the 50-byte answer category and other in the 250-byte category. The results of 50-bytes and 250-bytes run are described in Table 3.

5 Conclusion

We used various machine learning techniques for information retrieval in the context of TREC-9 filtering, Web, and QA tracks. These include BayesBoost for document filtering and a genetic algorithm for web-document retrieval. Though these approaches showed moderate performance this year, it is shown that they are very promising. Further improvement seems to require advanced preprocessing techniques as well as refinement in learning algorithms.

Acknowledgements

This research was supported by Korea Ministry of Information and Telecommunication under Grant 00-034 through IITA.

References

[1] Broglio, J., Callan, J.P., Croft, W.B., and Nachbar, D.W., Document Retrieval and Routing Using The INQUERY System, *The Third Text REtrieval Conference (TREC-3)*, pp. 29-38, 1995.

[2] Robertson, S. E. *et al*, Okapi at TREC-3, *The Third Text REtrieval Conference (TREC-3)*, pp. 109-126, 1995.

[3] Schapire, R. E. and Singer, Y, Improved Boosting Algorithms Using Confidence-rated Predictions, *Machine Learning 37(3)*, pp. 297-336, 1999.

[4] Schapire, R. E., Singer, Y., and Singhal, A., Boosting and Rocchio Applied to Text Filtering, In *Proc. SIGIR-98*, pp. 251-223, 1998.

[5] Shin, D-H. and Zhang, B-T., A Two-Stage Retrieval Model for the TREC-7 Ad Hoc Task, *The Seventh Text Retrieval Conference (TREC-7)*, pp. 501-508, 1998.

[6] Kim, S. and Zhang, B-T., Web-Document Retrieval by Genetic Learning of Importance Factors for HTML Tags, *Proceedings of PRICAI 2000 Workshop on Text and Web Mining*, pp. 13-23, 2000.

[7] Kim, Y-H., Hahn, S-Y., and Zhang, B-T. Text Filtering by Boosting Naive Bayes, In *Proc. SIGIR-00*, pp. 168-175, 2000.

[8] Kim, S.-D., Zhang, B-T., and Kim Y-T. Reducing Parsing Complexity by Intra-Sentence Segmentation Using Genetic Learning, *Machine Translation*, 2000 (to appear).