IMPROVED LATTICE-BASED SPOKEN DOCUMENT RETRIEVAL BY DIRECTLY LEARNING FROM THE EVALUATION MEASURES

Chao-hong Meng[1], *Hung-yi Lee*[2], *Lin-shan Lee*[1,2]

[1]Graduate Institute of Computer Science and Information Engineering, National Taiwan University
[2]Graduate Institute of Communication Engineering, National Taiwan University
{mno2, tlkagk}@speech.ee.ntu.edu.tw, lslee@gate.sinica.edu.tw

ABSTRACT

Lattice-based approaches have been widely used in spoken document retrieval to handle the speech recognition uncertainty and errors. Position Specific Posterior Lattices (PSPL) and Confusion Network (CN) are good examples. It is therefore interesting to derive improved model for spoken document retrieval by properly integrating different versions of lattice-based approaches in order to achieve better performance.

In this paper we borrow the framework of 'learning to rank' from text document retrieval and try to integrate it into the scenario of lattice-based spoken document retrieval. Two approaches are considered here, AdaRank and SVM-map. With these approaches, we are able to learn and derived improved models using different versions of PSPL/CN. Preliminary experiments with broadcast news in Mandarin Chinese showed significant improvements.

Index Terms— Spoken Document Retrieval, SVM-map, AdaRank, PSPL, Confusion Network

1. INTRODUCTION

With the rapid increase of multimedia content over the Internet, which often carries speech as the core information, the demand for spoken document retrieval has been growing very fast in recent years. Due to the inevitable uncertainty and errors in speech recognition especially under adverse environments, most state-of-the-art spoken document indexing methods have been based on multiple alternatives of recognition output, probably with some efficient representation. Good examples include Position Specific Posterior Lattices (PSPL) [1, 2], Confusion Network [3] and other similar variations. They are referred to as lattice-based approaches in this paper. Also, subword units have been widely used in spoken document retrieval to handle OOV and rare words as well as erroneous transcriptions. It has been found [2, 4] that different lattice-based approaches based on words or different subword units usually offer different performance in different situations, and proper combination of some of them is usually helpful. For example, in Mandarin Chinese it was found the linear combination of word-based and character-based PSPL scores is better than either case alone. It is therefore reasonable to try to derive an improved model by properly integrating scores from the various lattice-based approaches (PSPLs, CNs, etc.) based on words and various subword units in order to achieve better retrieval results.

On the other hand, in text document retrieval area the concept of 'learning to rank' have been successfully applied to derive better retrieval approaches. For example, the approaches using ensemble learning [5] and discriminative model [6]. As an example, Jun Xu et al. proposed a learning algorithm for ranking on the basis of boosting, referred to as AdaRank [7]. Yisong Yue et al. also performed the learning using Support Vector Machines, referred to as SVM-map [8]. These approaches are different from other previously proposed methods such as RankBoost [5] and Ranking SVM [9], because they tried to optimize some specially designed loss functions, which are directly related to the desired performance measures for retrieval. But these approaches were developed for text document retrieval, quite different from the scenario of spoken document retrieval considered here.

In this paper, we try to integrate the AdaRank [7] and SVM-map [8] learning algorithm into the framework of lattice-based spoken document retrieval approaches such as those based on PSPL and CN. Specifically, we use AdaRank and SVM-map learning algorithms to learn the weights for the scores generated from a variety of PSPL/CN-related indexing methods. Very encouraging improvements in performance were obtained.

2. GENERAL FRAMEWORK

Here the spoken document retrieval problem (in testing) is defined as, given any query, returning a list of desired documents in descending order of the relevance scores calculated with a ranking function. The objective of learning is then to construct such a ranking function (in training) based on some criteria. Such a ranking function have an input space $\mathcal{X}$ and an output space $\mathcal{Y}$.

Let the input space $\mathcal{X}$ be the set of all possible queries, or an infinite set $\mathcal{X} = \{q_1, q_2 \cdots\}$, where q_i is a query. The output space $\mathcal{Y}$, on the otherhand, consists of all possible ranks over a corpus (a rank denotes a permutation of all the documents in the corpus), $\mathcal{Y} = \{r_1, r_2, \cdots, r_l\}$, where each element in $\mathcal{Y}$ is a rank and l denotes the number of ranks. Assume there exists a total order among the ranks, $r_1 \succ r_2 \succ \cdots \succ r_{l-1} \succ r_l$, where '$\succ$' denotes a preference relationship, or r_1 is preferred than r_2 and so on.

In training, a set of training queries $\mathcal{Q} = \{q_1, q_2 \cdots, q_m\}$ is given. Each query q_i is associated with a list of retrieved documents $\mathbf{d_i} = \{d_{i1}, d_{i2}, \cdots, d_{i,n(q_i)}\}$ obtained with a specific retrieval approach (or ranker), and a list of labels $\mathbf{y_i} = \{y_{i1}, y_{i2}, \cdots, y_{i,n(q_i)}\}$, where $n(q_i)$ denotes the size of the lists $\mathbf{d_i}$ and $\mathbf{y_i}$, d_{ij} denotes the j^{th} document in $\mathbf{d_i}$, and y_{ij} denotes the order of document d_{ij} in the reference (or true) rank labels $\mathbf{y_i}$, $y_{ij} \in \{1, 2, \cdots, n(q_i)\}$. Thus the training set can be represented as $S = \{(q_i, \mathbf{d_i}, \mathbf{y_i})\}_{i=1}^{m}$. On the other hand, a feature vector $\vec{x_{ij}} = \Phi(q_i, d_{ij})$ can be created for each query-document pair (q_i, d_{ij}),$i = 1, 2, \cdots, m$; $j = 1, 2, \cdots, n(q_i)$, including many different relevant scores for (q_i, d_{ij}) obtained from many different retrieval approaches (rankers) as many features [5].

With the above, the goal here is to learn a function $f : \mathcal{X} \rightarrow \mathcal{Y}$. For any given query q_i, the output $f(q_i)$ is a rank, which is a permutation or ordered list of retrieved documents $\pi(q_i, \mathbf{d_i}, f)$, where $\mathbf{d_i}$ is the correspoding list of $n(q_i)$ retrieved documents. The goal of learning here is to minimize a loss function representing the disagreement between the output permutation $\pi(q_i, \mathbf{d_i}, f)$ and the reference (or true) rank $\mathbf{y_i}$ for $\mathbf{d_i}$, for all queries. Such a loss function may be expressed by $\Delta : \mathcal{Y} \times \mathcal{Y} \rightarrow \mathbb{R}$, where $\Delta(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f))$ quantifies the loss for the output $\pi(q_i, \mathbf{d_i}, f)$ if the correct rank is $\mathbf{y_i}$. The loss function used in the work here is "1 - $E(y_i, \pi(q_i, \mathbf{d_i}, f))$", where $E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f))$ is the performance measure evaluated for $\mathbf{y_i}$ and $\pi(q_i, \mathbf{d_i}, f)$. Here $E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f))$ is taken as the mean average precision (MAP). The upper bound of this loss function will be directly minimized below by two learning algorithms: AdaRank and SVM-map.

Table 1. Notations and explanations

Notations	Explanations
$q_i \in Q$	i^{th} training query
$\mathbf{d_i} = \{d_{i1}, d_{i2}, \cdots, d_{i,n(q_i)}\}$	List of retrieved documents for q_i
$\mathbf{y_i} = \{y_{i1}, y_{i2}, \cdots, y_{i,n(q_i)}\}$	Reference (true) rank list for q_i and $\mathbf{d_i}$
$S = \{(q_i, \mathbf{d_i}, \mathbf{y_i})\}_{i=1}^m$	Training set
$\pi(q_i, \mathbf{d_i}, f)$	Permutation of document in $\mathbf{d_i}$ given q_i returned by f
$\Delta(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f))$	Loss function
$E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f)) \in [-1, +1]$	Performance Measure
$\Phi(q_i, d_{ij})$	Function to extract features

3. ADARANK

Based on the concept of AdaBoost, AdaRank [7] constructs the ranking function by integrating a group of weak rankers. It is adaptive in the sense that, in each iteration, the subsequent ranker constructed are tweaked in favor of those instances misranked by the previous rankers. As mentioned above, AdaRank here directly optimize the upper bound of the loss function "1 - $E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f))$", or "1 - MAP" in this work. The complete algorithm is summarized here [7]. Given a training set $S = \{(q_i, \mathbf{d_i}, \mathbf{y_i})\}_{i=1}^m$, we first set the number of iterations T. In each iteration t AdaRank then creates a weak ranker $h_t (t = 1, \cdots, T)$. At the end of iteration T, it outputs a ranking function f_T by linearly combining all the weak rankers obtained.

At each iteration t, AdaRank maintains a weight distribution $\mathbf{P_t}(i)$ over all the queries q_i in the training set. This distribution plays the same role as the weight distribution of AdaBoost. Initially, AdaRank sets equal weights to all the queries, or $\mathbf{P_1}(i) = \frac{1}{m}$. At each iteration t, it increases the weights for those queries that are not ranked well by the function created so far, or f_t. As a result, the learning in the next iteration $t + 1$ will be focused on the creation of a weak ranker that can work on the ranking of those 'hard' queries. At each iteration t, a weak ranker h_t is constructed based on the training set with the weight distribution $\mathbf{P_t}(i)$. In this paper, we consider the scores of various lattice-based approaches using words or various subword units as candidates,

and select the one giving the best weighted performance over all queries as the weak ranker constructed at iteration t.

$$h_t = \arg\max_{x_k} \sum_{i=1}^{m} P_t(i) E(\mathbf{y_i}, \pi(q_i, d_i, \mathbf{x_k})) \,, \tag{1}$$

where $\mathbf{x_k}$ denotes a ranking function using the score of the k^{th} lattice-based indexing methods,

Once a weak ranker h_t is found, AdaRank chooses a weight $\alpha_t > 0$ for the weak ranker [7]. Intuitively, α_t measures the importance of h_t. A ranking function f_t is then created at each iteration t by linearly combining the weak rankers obtained so far $h_1, \cdots, h_t$ with weights $\alpha_1, \cdots, \alpha_t$, f_t is then used for updating the distribution $\mathbf{P_{t+1}}(i)$ (we made some modifications on the original formula here). The complete algorithm is listed below.

Algorithm 1 AdaRank Learning Algorithm

1: Input: S = $\{(q_i, d_i, y_i)\}_{i=1}^m$ and parameters E and T
2: Initialize $P_1(i) = \frac{1}{m}$.
3: **for** $t = 1, \cdots, T$ **do**
4: Create weak ranker h_t using Eq(1).
5: Choose α_t
$$\alpha_t = \frac{1}{2} \cdot \frac{\sum_{i=1}^m \mathbf{P_t}(i)\{1 + E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, h_t))\}}{\sum_{i=1}^m \mathbf{P_t}(i)\{1 - E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, h_t))\}}$$
6: Create f_t
$$f_t(q) = \sum_{k=1}^{t} \alpha_k h_k(q)$$
7: Update $\mathbf{P_{t+1}}$
$$\mathbf{P_{t+1}}(i) = \frac{\mathbf{P_t}(i) exp\{-E(\mathbf{y_i}, \pi(q_i, \mathbf{d_i}, f_t))\}}{\sum_{j=1}^m \mathbf{P_t}(j) exp\{-E(\mathbf{y_j}, \pi(q_i, \mathbf{d_i}, f_t))\}}$$
8: **end for**
9: Output ranking model: $f(q) = f_T(q)$.

4. SVM-MAP

The goal here is to rank the documents by their relvance to the query. A similarity function comparing different ranks was proposed ealier [7, 8], which is helpful in achieving this goal using Support Vector Machine (SVM). With this similarity function, we can learn a discriminative function (a hyper plane in SVM) to separate the best rank from other ranks in the output space $\mathcal{Y}$. In this way, the algorithm SVM-map [8] is not only armed with the power of SVM, but able to directly optimize the upper bound of the loss function here, 1 - MAP. Some other loss functions were also discussed [10].

The goal of SVM-map is to learn a vector $\mathbf{w}$ consisting of the weights for different features in the feature vector $\vec{x_{ij}}$ extracted by $\Phi(q_i, d_{ij})$. In this way, the output of the ranking function for a query q_i, $f(q_i)$ is simply a rank based on the relevance scores $\mathbf{w} \cdot \vec{x_{ij}}$. In the work here, we use the scores from the various lattice-based appraoches based on words or various subword units as the features in $\vec{x_{ij}}$. We first define the similarity function $F(\mathbf{r}, q_i, \mathbf{d_i}; \mathbf{w})$ as the similarity between any rank $\mathbf{r}$ in the output space $\mathcal{Y}$ and the permutation obtained with the weight vector $\mathbf{w}$ given query q_i, list of retrieved documents $\mathbf{d_i}$ and the relevance score $\mathbf{w} \cdot \vec{x_{ij}}$. The optimization problem of SVM-map can then be written as follows:

$$\min_{\mathbf{w}, \xi_i \geq \mathbf{0}} \{\frac{1}{2}\|\mathbf{w}\|^2 + \frac{C}{m} \sum_{i=1}^{m} \xi_i\}, \tag{2}$$

s.t. $\forall i, \forall \mathbf{r} \in (\mathcal{Y} - \mathbf{y_i})$

$$F(\mathbf{y_i}, q_i, \mathbf{d_i}; \mathbf{w}) \geq F(\mathbf{r}, q_i, \mathbf{d_i}; \mathbf{w}) + \Delta(\mathbf{y_i}, \mathbf{r}) - \xi_i \,,$$

where $\mathbf{y_i}$ is the reference (true) rank, $\mathbf{r}$ is any other rank, $\Delta(\cdot, \cdot)$ is the loss function mentioned previously, and the ξ_i and C are the slack variables and the trade-off parameter of SVM [7].

The above optimization problem can be considered as minimizing the following:

$$\sum_i \max_{\mathbf{r}\in\mathcal{Y}-\mathbf{y_i}} (\Delta(\mathbf{y_i},\mathbf{r}) - (F(\mathbf{y_i},q_i,\mathbf{d_i};\mathbf{w}) - F(\mathbf{r},q_i,\mathbf{d_i};\mathbf{w}))) + \mu\|\mathbf{w}\|^2 \quad (3)$$

Minimizing Equation (3) above actually means that we try to minimize the loss function $\Delta(\mathbf{y_i},\mathbf{r})$ while maximizing the margin between the similarity functions for the true rank $\mathbf{y_i}$ and other ranks $\mathbf{r}$. The term $\mu\|\mathbf{w}\|^2$ is included in all SVM formulation. Equation (3) was also proved as the upper bound of "1 - MAP" [11], so as long as we minimize Equation (3), the rank based on the relevance score $\mathbf{w}\cdot\vec{x_{ij}}$ is the best rank.

Although the optimization problem in Equation (2) has a huge number of constraints, an approximate solution can be found with an algorithm by selecting a set of active constraints from all the constraints [8, 10]. In this algorithm, the most violated constraint is found in each iteration to join an active constraint set, until no more new constraints can be found to join the active constraint set.

5. EXPERIMENT

5.1. Experimental Setting

The corpus used in the experiments were the Mandarin broadcast news stories collected daily from local radio stations in Taiwan from August to September 2001. We manually divided these stories into 5034 segments, each with one to three utterances and taken as a seperate document to be retrieved. From the bigram lattices of these segments we generated the corresponding word-based PSPL/CN and subword-based PSPL/CN using Chinese characters and Mandarin syllables. By altering the beam width in generating the bigram lattice, four lattices L_1, L_2, L_3 and L_4 were generated, each with averaged 19.89, 30.27, 46.75 and 72.77 edges per spoken word respectively, from which PSPL/CN of different depths and sizes based on words, characters and syllables were obtained. A lexicon of 62K words was used. The acoustic models included a total of 151 intra-syllable right-context-dependent Initial-Final (I-F) models trained with 8 hours of broadcast news stories collected in 2000. The recognition character accuracy obtained for the 5034 segments was 75.27% (under trigram one-pass decoding).

200 text test queries were generated by manual selection from a set of automatically generated candidates. The candidates were patterns of 1-3 words which appeared at least 5 times in the 5034 segments. 44 of the 200 queries included OOV words and categorized as OOV queries, while the remaining 156 were in-vocabulary (IV) queries.

We implemented the AdaRank learning algorithm for AdaRank, and used the SVM^{map} developed by Yisong Yue and Thomas Finley [12] for SVM-map. Tradeoff parameter of SVM-map was empirically set to 10^4. AdaRank requires different indexing methods as the weak rankers, and SVM-map requires the scores of weak rankers as features. We thus implemented word-based, character-based and syllable-based PSPL/CN (3x2), each with the accumulated probability of uni-, bi- and tri-grams respectively (3), so a total of 18 (6x3) different indexing methods.

All retrieval results presented here are calculated in terms of Mean Average Percision (MAP) evaluated with the standard trec_eval package used by the TREC evaluations. Both AdaRank and SVM-map need queries to train the ranking function. Considering the limited number of available queries here, we performed

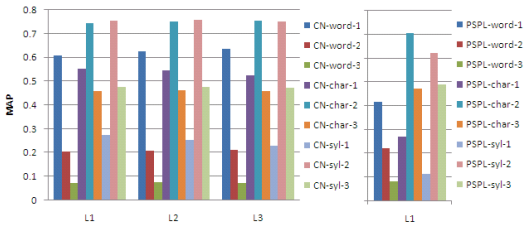

Fig. 1. Ranking accuracies of the 18 indexing methods (weak rankers) by CN(left, $L_1 - L_3$) and PSPL(right, L_1) when used alone

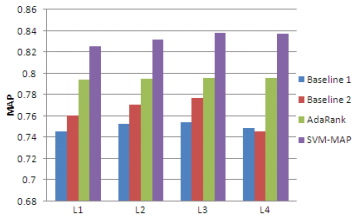

Fig. 2. Ranking accuracies of the two baselines, AdaRank and SVM-map

4-fold cross validation on the 200 text test queries in the AdaRank and SVM-map experiments (i.e., training with 150 queries and testing with 50 queries, repeating four times and averaging the results.)

5.2. Experimental Results

We first evaluate the MAP score for each of the 18 indexing method (weak rankers) alone, and show the results in Figure 1, where the left part show the 9 CN-based methods for three lattice sizes L_1, L_2, L_3. It can be found the trend is very consistent and therefore the results for L_4 is left out here. On the right part of Figure 1 is the 9 PSPL-based methods for L_1 alone. The trend is still very similar to that of CN-based methods, although slightly different. From this figure, we can see that character-based bi-gram of both CN and PSPL (CN-char-2, PSPL-char-2), and syllable-based bi-gram of both CN and PSPL (CN-syl-2, PSPL-syl-2) are the top 4 when a single indexing method is considered. This is reasonable because the majority of words in Chinese includes two mono-syllabic characters, or have two characters pronounced as two syllables. Although many homonym characters may share the same syllable, very small number of homonym bi-character words share the same bi-syllablic pattern. So the bi-gram of these two subword units (characters/syllables) carry plenty of information.

The results of AdaRank and SVM-map are shown in Figure 2, for the four lattice sizes L_1-L_4. We use 2 baselines here as first two bars in each set: baseline 1 is the best single indexing method (weak ranker) out of the 18 mentioned above, and baseline 2 is the uniform integration of the 18 weak rankers, each weighted by $\frac{1}{18}$. The results of AdaRank and SVM-map are shown as the 3-rd and 4-th bars in each set in Figure 2. We see in general baseline 2 is better than baseline 1 (except for L_4, in which two large lattice size may produce too many noisy features, but AdaRank is always better, and SVM-map is high above, roughly 6% absolute higher than baseline 2. The power of SVM was verified here.

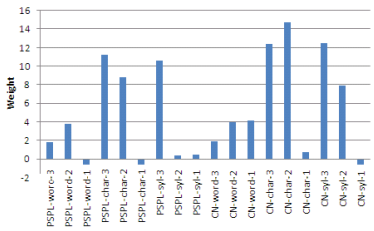

Fig. 3. weight vector $\mathbf{w}$ learned by SVM-map for L_1 lattice size and fold-1

5.3. Further Analysis

Since SVM-map achieved the best results, we further analyze the weights learned by the SVM-map here. The weights learned by L1-fold1 (out of the 4-fold cross validation) are plotted in Figure 3 (the results are very similar for other lattice sizes and folds). We can see that the highest weight went to the character-based bigram of CN (CN-char-2), the best indexing method in Figure 1 if considered alone. The next 4 highest weights went to syllable-based and character-based trigram of both CN and PSPL. First, since many homonym characters with different meanings may share the same syllable in Mandarin Chinese, clearly characters carry more precise information than syllables. This is why character-based bi-gram was chosen to give the highest weight, out of the 4 subword unit based bi-grams which are best if considered alone in Figure 1. Once it is chosen, the other 3 becomes less important since they carry similar information. Also, although characters carry more precise information than syllables, syllables have much higher recognition accuracies than characters (since many characters may share the same syllable). As a result the information carried in syllables and characters actually complement each other. Also the subword based trigrams very often extend beyond a word (the majority of Chinese words are bi-syllabic or bi-character) which carry very useful information. This is probably why the next 4 highest weights went to the 4 subword-based tri-grams.

It is interesting to note that there are three negative weights in Figure 3. They are for word, character and syllable uni-grams respectively. The MAP score would be degraded if these three indexing methods were removed. So they made positive contributions even if their weights are negative. One possible reason may be that many of the bi- and tri-gram indexing methods have their indexing features overlapping with each other, so many features were counted repeatedly. These negatively weighted uni-grams then probably reduced such repetitions. Note that uni-grams may be much better tools for reducing such repetitions than bi- or tri-grams.

As another analysis, we arbitrarily chose 3 indexing methods out of the 18, combined them using the weights given by SVM-map to generate a total of $\binom{18}{3} = 816$ sub-optimal rankers, and list the top 5 sub-optimal rankers in Table 2 (for L_1 and fold-1). Note that with only 3 weak rankers they are already much higher than baseline 2 in Figure 2, which is the uniform combination of all the 18 weak rankers. Again this verifies the power of SVM-map.

Also, almost all of these top 5 sub-optimal rankers include a good combination of different appraoches: one syllable-based, one character-based, one word-based (for recognition accuracies, syllble $>$ character $>$ word, for semantic information, word $>$ character $>$ syllable); one or two CN-based and the other PSPL-based; uni-grams and tri-grams; units with lengths ranging from 1 syllable (syllable/character-based unigrams) up to 6 syllables (word-based tri-grams). Clearly the good combination of a variety of properties for the different indexing methods leads to the high performance. Interestingly, each of these top 5 sub-optimal rankers also includes a weak ranker with negative weight. Clearly the negative weights are important.

Table 2. The top 5 sub-optimal rankers by three different methods

Rank	Method 1	Method 2	Method 3	MAP
1	syl-CN-3	char-PSPL-1	word-PSPL-3	0.8166
2	syl-CN-3	char-PSPL-1	word-CN-3	0.8166
3	syl-CN-3	word-PSPL-1	word-PSPL-3	0.8161
4	syl-CN-3	word-PSPL-1	word-CN-3	0.8161
5	syl-CN-3	syl-CN-1	word-PSPL-3	0.8153

6. CONCLUSION

In this paper we borrow the frameword of 'learning to rank' from text document retrieval to be used with the lattice-based spoken document retrieval approaches. This framework enables us to automatically learn the proper weights when integrating different versions of lattice-based indexing methods based on words or different subword units. In preliminary experiments with broadcast news in Mandarin Chinese, not only significant improvements were obtained, but it was found that SVM-map learning algorithm always achieves better results than AdaRank learning algorithm.

7. REFERENCES

[1] Ciprian Chelba and Alex Acero, "Position specific posterior lattices for indexing speech," in *ACL '05: Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Ann Arbor, Michigan, 2005, pp. 443–450, Association for Computational Linguistics.

[2] Yicheng Pan and Berlin Chen, "Subword-based position specific posterior lattices (s-pspl) for indexing speech information," in *the 10th European Conference on Speech Communication and Technology (Interspeech - Eurospeech 2007)*, 2007, pp. 318–321.

[3] T. Hori, I.L. Hetherington, T.J. Hazen, and J.R. Glass, "Open-vocabulary spoken utterance retrieval using confusion networks," in *ICASSP 2007*, 2007, pp. 73 – 76.

[4] Yicheng Pan, Hung lin Chang, and Lin shan Lee, "Analytical comparison between position specific posterior lattices and confusion networs based on words and subword units for spoken documenbt indexing," in *the 10th IEEE Automatic Speech Recognition and Understanding Workshop (ASRU), Kyoto, Japan*, 2007, pp. 677–682.

[5] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer, "An efficient boosting algorithm for combining preferences," *J. Mach. Learn. Res.*, vol. 4, pp. 933–969, 2003.

[6] Ramesh Nallapati, "Discriminative models for information retrieval," in *SIGIR '04: Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval*, Sheffield, United Kingdom, 2004, pp. 64–71, ACM.

[7] Jun Xu and Hang Li, "Adarank: a boosting algorithm for information retrieval," in *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, 2007, pp. 391–398, ACM.

[8] Yisong Yue, Thomas Finley, Filip Radlinski, and Thorsten Joachims, "A support vector method for optimizing average precision," in *SIGIR '07: Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*, Amsterdam, The Netherlands, 2007, pp. 271–278, ACM.

[9] R. Herbrich, T. Graepel, and J.H. Friedman, "Large margin rank boundaries for ordinal regression," MIT Press, Cambridge, MA, 2000.

[10] Thorsten Joachims, "A support vector method for multivariate performance measures," in *ICML '05: Proceedings of the 22nd international conference on Machine learning*, Bonn, Germany, 2005, pp. 377–384, ACM.

[11] Jun Xu, Tie-Yan Liu, Min Lu, Hang Li, and Wei-Ying Ma, "Directly optimizing evaluation measures in learning to rank," in *SIGIR '08: Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, Singapore, Singapore, 2008, pp. 107–114, ACM.

[12] "http://projects.yisongyue.com/svmmap/," .