EFFICIENT INTERACTIVE RETRIEVAL OF SPOKEN DOCUMENTS WITH KEY TERMS RANKED BY REINFORCEMENT LEARNING

Yi-cheng Pan, Jia-yu Chen, Yen-shin Lee, Yi-sheng Fu, and Lin-shan Lee

Graduate Institute of Computer Science and Information Engineering, National Taiwan University
Taipei, Taiwan, Republic of China
speech@speech.ee.ntu.edu.tw

ABSTRACT

Unlike written documents, spoken documents are difficult to display on the screen; it is also difficult for users to browse these documents during retrieval. It has been proposed recently to use interactive multi-modal dialogues to help the user navigate through a spoken document archive to retrieve the desired documents. This interaction is based on a topic hierarchy constructed by the key terms extracted from the retrieved spoken documents. In this paper, the efficiency of the user interaction in such a system is further improved by a key term ranking algorithm using Reinforcement Learning with simulated users. Significant improvements in retrieval efficiency, which are relatively robust to the speech recognition errors, are observed in preliminary evaluations.

Index Terms: Interactive Spoken Document Retrieval, Key Term Hierarchy, Ranking Algorithm.

1. INTRODUCTION

In a recent paper, we proposed using multi-modal dialogues to help the user "navigate" across a spoken document archive to retrieve the desired documents [1]. For a given user query, the retrieval system produces a topic hierarchy constructed from the retrieved spoken documents to be shown on the screen of the hand-held devices. The user can then expand his query easily by choosing the key terms within the topic hierarchy by a simple click or an additional spoken query to specify more clearly what he is looking for. With a few dialogue turns, the small set of spoken documents desired by the user can be found by a more specific query expanded during the dialogue process.

Given the user's initial query, however, a large number of key terms are usually extracted and the constructed topic hierarchy is very often too large to be shown on the screen of a hand-held device. In this paper, we further propose a new ranking algorithm for the key terms based on Reinforcement Learning [2, 3, 4, 5] considering both the semantic structure of the document archive and the information needs of the user. It is shown that training the ranking algorithm directly based on the search efficiency and user satisfaction offers improved retrieval efficiency over conventional term ranking methods used in the area of information retrieval. In the experiments, we also show that the proposed ranking algorithm is robust against recognition errors.

2. KEY TERM RANKING BY REINFORCEMENT LEARNING WITH SIMULATED USERS

In the proposed approach for spoken document retrieval/navigation (SDR), the input user query is usually very short and the retrieved documents very numerous. A large number of key terms are usually extracted and the constructed topic hierarchy hence tends to be too large. It is therefore crucial to have a good strategy for ranking all of these key terms and for moving these important key terms towards the top of each layer of the hierarchy, so as to make it easier for the user to choose, and thus to produce efficient interaction between the user and the system. Such a term ranking strategy has to achieve a good balance between two mutually contradicting factors of the key terms — i.e., *coverage* (more documents can be retrieved with the key term) and *discriminating ability* (the key term indicates a more specific topic against other irrelevant documents) — by reflecting the semantic structure of the document archive. The approach proposed here for the above purpose is based on the well known concept of Reinforcement Learning [2, 3, 4, 5]. In this approach, the training data used are simply those generated by simulated users.

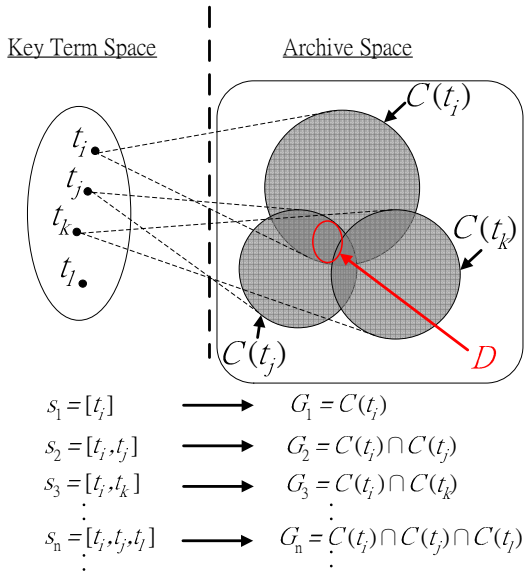

Fig. 1. Key term space and archive space.

2.1. Key term space and archive space

As shown in the left half of Fig. 1, all the key terms t_i, t_j, t_k, t_l, ... for a given document archive form a key term space. Every key term t_i can be mapped to a class of documents $C(t_i)$ which can be retrieved by the term t_i with a given retrieval approach in the archive space as shown in the right half of Fig. 1. Thus the mapping relation

between the key term t_i and the corresponding document class $C(t_i)$ is defined by the given retrieval approach, and the formulation here is equally applicable to all retrieval approaches, including the Vector Space Model, Latent Semantic Indexing (LSI) [6], or Probabilistic Latent Semantic Indexing (PLSI) [7].

The user may initiate the retrieval process by entering the first key term t_i as the query, which is referred to as the first state s_1; the retrieved results are the group of documents G_1, or

$$s_1 = [t_i] \longrightarrow G_1 = C(t_i). \quad (1)$$

When G_1 is very large, the user may further enter the next key term t_j or t_k to expand his query, respectively referred to as states s_2 or s_3, and the retrieved results are denoted as G_2 or G_3,

$$s_2 = [t_i, t_j] \longrightarrow G_2 = [C(t_i)] \cap [C(t_j)], \quad (2)$$

$$s_3 = [t_i, t_k] \longrightarrow G_3 = [C(t_i)] \cap [C(t_k)], \quad (3)$$

and so on. This process can continue: for example, the query for state s_2 above can be further expanded by a third key term t_l to produce another state s_n with the retrieved results G_n,

$$s_n = [t_i, t_j, t_l] \longrightarrow G_n = [C(t_i)] \cap [C(t_j)] \cap [C(t_l)]. \quad (4)$$

2.2. Reinforcement Learning by Simulated Users

In the reinforcement learning algorithm, we need a huge quantity of training data, which are generated by simulated users as described below.

2.2.1. Simulating the User's Information Needs

Here we simulate the information needs of the user, which is a set of desired documents, *D*, in the archive space. In observing real users' information needs, we found two characteristics: 1) the documents in *D* have similar semantics to each other, and 2) the documents in *D* usually share common key terms. So we first cluster the whole document archive into *C* clusters. *C* should be chosen such that each cluster has enough documents[1]. Then we randomly pick one cluster and extract all of the key terms from the documents in that cluster. A key term $\bar{t}$ is then randomly chosen from the extracted key terms, and the set of documents including this key term $\bar{t}$ is denoted as *D*'. A random number M is generated as the size of the desired document set *D*. If the number of documents in *D*' exceeds M, exactly M documents are randomly selected and the set of them is regarded as D. Otherwise, we choose another key term most similar to $\bar{t}$ but not yet chosen and add documents including this key term into *D*' until the number of documents in *D*' equals or exceeds M. The similarity used here is based on the probability distribution of each key term on the latent topics in the same Probabilistic Latent Semantic Analysis (PLSA) model used in the PLSI retrieval process[2]. For example, the key term "White House" may result in the most similar key terms "U.S.", "George Bush", "Middle East", and so on. The set of desired documents *D* generated in this way in general satisfies the two requirements mentioned above.

2.2.2. Simulating the Retrieval Behavior of a User

With the desired document set D generated as above for a simulated user, we then simulate the user's retrieval behavior as follows. A key term t_i which can retrieve at least some documents in D is randomly selected as the first query, or state s_1, for this simulated user, and the group of documents $G_1 = C(t_i)$ is obtained as in relation (1). Very possibly G_1 is much larger than the desired document set D, so the simulated user may enter the next key term t_j or t_k, and so on. In general, t_j or t_k can be any key terms that can retrieve at least some documents in $C(t_i)$. Each of these possible key terms is then simulated as $s_2 = [t_i, t_j]$, $s_3 = [t_i, t_k]$, producing the retrieved results G_2, G_3 and so on as in relations (2)(3). This process can continue: for example, entering the third key term t_l after $s_2 = [t_i, t_j]$ as state s_n=$[t_i, t_j, t_l]$ as in relation (4). Assume the user is satisfied when the recall rate r is above a threshold r_0, i.e., L out of the top K retrieved documents are within the desired document set D, or $r = L/M > r_0$, where M is the number of documents in D.

The above simulation results for a simulated user with a desired document set D and an initial key term t_i can be represented by a tree of all the states as defined in relations (1)(2)(3)(4), and a typical example of such a tree is partially shown in Fig. 2. For example, after the first key term is entered as state s_1, different choices of the second key term result in the different states s_2, s_3 and s_4 on the next layer, and so on. The leaf nodes of the tree represented by double circles are the final states — or those states where the user is satisfied — labeled by a score $m(\cdot)$ which is the number of key terms successively entered, or the number of steps required to arrive at the final state along the path from the root. Clearly, the smaller $m(\cdot)$ the better. After the tree is constructed, backtracking from each leaf node along a path back to the root gives a score u to each intermediate state, which is the minimum score $m(\cdot)$ for all child leaf nodes of the intermediate state as below,

$$u = \min_i [m(s_i)], \quad (5)$$

where the minimization is performed over all child leaf nodes of the state. For example, in Fig. 2 $u = 3$ for s_2 because $m(s_6) = 3$, but $u = 4$ for s_5 because $m(s_8) = 4$. For a given parent node, the next-layer child node with smaller score u is clearly better, because the user can be satisfied earlier by following that key term. For example, for the three states s_2, s_3, s_4 in the second layer in Fig. 2, s_4 is better than s_2, and s_2 is in turn better than s_3. Note that in order to avoid excessively lengthy path expansions in the trees we need to prune the tree branch if no documents can be retrieved at the branching state.

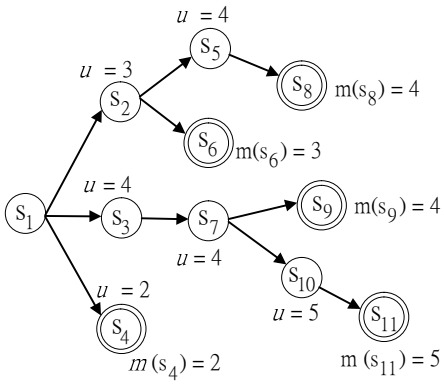

Fig. 2. A typical tree constructed for the retrieval states for a simulated user.

[1] In our system, *C* was set as 12.

[2] In our system, we used 350 latent topics.

2.3. The Proposed Ranking Algorithm

It should be noted that given the documents archive and a set of key terms, the document classes $C(t_i)$ for all key terms t_i are fixed, as are the intersection relationships among them such as G_2, G_3, G_4 in relations (2)(3)(4). As a result, when starting with a certain state $s_1 = [t_i]$ as in relation (1), all the possible next key terms such as t_j, t_k in relations (2)(3) are also fixed, as are the next layer states in the tree in Fig.2. In other words, the fundamental structure of the tree in Fig. 2 given the root s_1 is fixed, but the specific *realization* of the tree is dependent on the desired document set D; that is, different desired document sets D will in effect prune the branches of the fundamental tree structure at different layers, thereby producing a different set of leaf nodes for each D.

As mentioned previously, every simulated user is defined by a simulated desired document set D and an initial query key term t_i, followed by all possible following key terms t_j, t_k and so on. The initially large number of users can be further classified into categories by the initial query key term t_i. In each category all the simulated users start with the same initial query key term t_i and the same root state $s_1 = [t_i]$, and thus have the same basic tree structure, but the different desired document sets D for the different simulated users produce different leaf nodes and different tree realizations, which result in different scores as in equation (5) for each state in the basic tree structure. All different scores u for the different tree realizations for a certain state s_j in the basic tree structure can then be averaged over all of the tree realizations to give an averaged score $\bar{u}$ for the state s_j. This produces a single tree for a category of users starting with the same initial key term, t_i. This tree describes the retrieval efficiency of all of the possible expansions of the initial query, which can be seen as the "averaged tree" for an initial key term t_i.

The set of all "averaged trees" as mentioned above for all initial key terms t_i is then the optimized ranking formula. It reflects the semantic structure of the document archive, and indicates how efficient each key term is in identifying the desired documents given the previously entered queries, which constitute a key term set; that is, a state. This efficiency represents an automatic balance between two mutually contradicting factors for the key terms as mentioned previously: coverage and discriminating ability. Such a balance is obtained by averaging the retrieval processes of a large number of simulated users.

Taking into consideration the SDR problem, it can be seen that after the real user enters a query, a topic hierarchy is constructed for the many retrieved documents with nodes labeled by key terms. The many key terms on a level of the topic hierarchy are then sorted according to this ranking algorithm. In this way, the user can select the next key term from the top to expand his query and indicate his information need precisely and efficiently, even on a small screen, i.e., of a hand-held device.

3. EXPERIMENTAL RESULTS

An initial prototype system was successfully developed at National Taiwan University (NTU). Mandarin Chinese broadcast news segments were taken as the example spoken/multi-media documents. Here the broadcast news archive to be navigated across and retrieved from included 10,000 news stories, among which about 2000 key terms were extracted [8]. A total of 5,000,000 users were simulated in training the ranking algorithm.

We evaluated the performance of interactive retrieval in terms of task success rate and the number of key terms needed for a successful retrieval. The task is defined to be successful if the user is satisfied after entering a number of queries, as defined in Sec. 2.2.2, i.e., L out of the top K retrieved documents are within the desired document set D, and $L/M > r_0$, where M is the number of documents in D and r_0 is the desired recall rate. M was taken as a random number uniformly distributed in [5,100]. K was set to 25 and r_0 to 0.15. In the test process, before user satisfaction the test user is requested to follow the ranking list proposed by the system to see whether the current top ranking key term is among his information need, D. Once the first qualified key term is found, the user simply chooses the key term to augment his query. Another set of retrieved documents and another ranking of the respective key terms are then displayed to the user to judge the user's satisfaction or to proceed the next run.

We generated 1000 test users in the same way as we simulated the training users as presented in Secs. 2.2.1 and 2.2.2. Two graduate students of NTU's Graduate Institute of Journalism helped another 50 tests including the desired document set D and the entered queries. The proposed ranking algorithm is compared against two previously proposed algorithms, the *wpq* method proposed by Robertson [9] and the *tf-idf* method. All three methods were applied on the same set of key terms extracted from the spoken documents.

3.1. Results for the Ranking Algorithm by Simulated Testing users

The results tested on 1000 simulated testing users are listed in Table 1. In the first part (A), the speech recognition was assumed to be 100% correct, while the character recognition accuracies were assumed to degrade to lower numbers in parts (B)(C)(D). In cases (B)(C)(D), we simulated the recognition errors for each query both in the testing and the training set by generating feature vectors according to the Hidden Markov Models with increased Gaussian mixture variances. During training, simulated queries for each initial query were generated with random errors, which in turn generated trees slightly different from that for the initial query and so on. From part (A) of Table 1, we see that with the proposed ranking algorithm, the best task success rate of 89.2% can be achieved, as compared to the 78.6% and 33.1% for *wpq* method and *tf-idf* methods respectively. Also, the average number of key terms needed for successful trials is as low as 2.13 for the proposed ranking algorithm, as compared to 3.08 and 3.46 for *wpq* and *tf-idf* methods. Fig. 3 (a) shows the detailed numbers of failure trials and successful trials completed in different number of key terms out of the 1000 simulated testing users for the cases listed in part (A) of Table 1.

It was found that with the *tf-idf* method, 669 out of the 1000 trials failed; all successful trials were finished within 7 turns. Much better performance was obtained for the *wpq* method. However, when the proposed ranking algorithm was used, only 108 trials failed, and all trials were completed within 3 turns, a result significantly better even than the *wpq* method. Similar situations as in part (A) of Table 1 can be observed in parts (B)(C)(D) of Table 1. For example, when the character accuracy is 74% in part (D), the proposed approach still achieved a success rate of 80.1%, as compared to the serious degradations in the other approaches, because in those cases the recognition errors in the queries simply led to totally irrelevant key terms. Similar plots for the detailed numbers of this case in part (D) can be seen in Fig. 3 (b).

In Fig. 4 we plot the task success rate and average number of key terms needed in successful trials for the three ranking methods as functions of the recognition accuracies. It can be found that the performance of the proposed ranking algorithm is quite robust with respect to recognition errors, while the other two methods are quite

sensitive to recognition errors.

	ACC	Algorithms	R_{sucess}	$N_{key\ term}$
without Recognition Errors	(A) 100%	*Proposed*	89.2%	2.13
		wpq	78.6%	3.08
		tf-idf	33.1%	3.46
with Recognition Errors	(B) 92%	*Proposed*	89.1%	2.21
		wpq	75.2%	4.33
		tf-idf	30.1%	5.78
	(C) 88%	*Proposed*	86.0%	2.33
		wpq	70.5%	4.89
		tf-idf	21.0%	6.01
	(D) 74%	*Proposed*	80.1%	3.34
		wpq	55.2%	5.23
		tf-idf	11.2%	6.31

Table 1. The results for the proposed ranking algorithm compared with the *wpq* and *tf-idf* methods, without and with recognition errors, tested by 1000 simulated testing users. $R_{success}$ is the task success rate and $N_{key\ term}$ the average number of key terms needed in a successful retrieval.

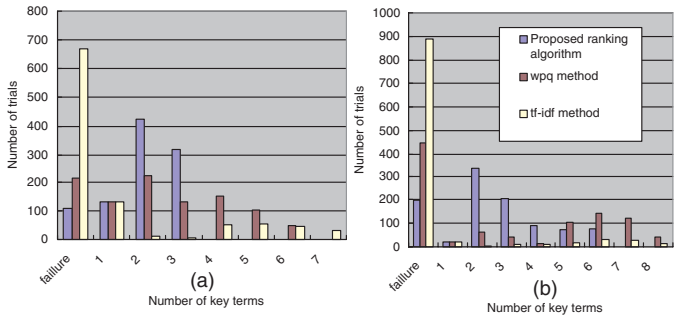

Fig. 3. Number of failure trials and successful trials completed in different number of tkey terms for the proposed ranking algorithm compared to the *wpq* and *tf-idf* methods. (a) for part (A) and (b) for part (D) in Table 1.

3.2. Results for the Ranking Algorithm by Real Users

Here we performed 50 tests handcrafted by two graduate students of NTU's Graduate Institute of Journalism with their desired document sets D and their initial queries, and the results are listed in Table 2. It can be found that with human testers, a similar situation to that mentioned above can be observed.

Experiments	$R_{success}$	$N_{key\ term}$
Proposed ranking algorithms	81.0%	2.62
wpq method	79.0%	3.10
tf-idf method	28.0%	3.54

Table 2. Experimental results for the different ranking algorithms, tested by human users.

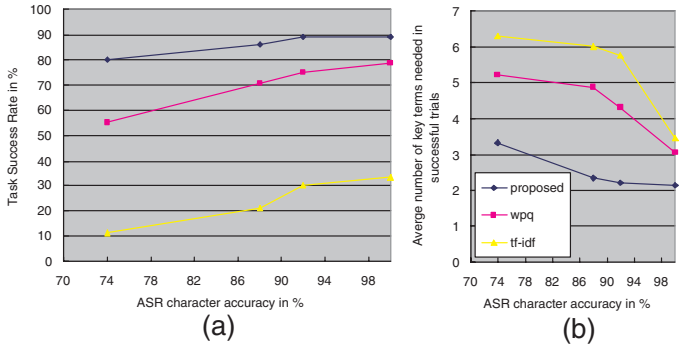

Fig. 4. (a) the task success rates and (b) the average numbers of key terms needed in successful trials for different recognition accuracies for the three ranking methods for the four cases in Table 1.

4. CONCLUSION

In this paper we proposed using Reinforcement Learning to rank key terms to guide the user more efficiently. In Reinforcement Learning we proposed using simulated users as training data, and asserted that learned key term ranking can reflect the semantic structure of the document archive. Experiments showed that the proposed ranking algorithm offered much better performance than convetnional methods and is relatively robust to recognition errors.

5. REFERENCES

[1] Y.-C. Pan, C.-C Wang, Y.-C Hsieh, T.-H Lee, Y.-S Lee, Y.-S Fu, Y.-T Huang, and L.-S Lee, "A multi-modal dialogue system for information navigation and retrieval across spoken document archives with topic hierarchies," in *ASRU*, 2005, pp. 375–380.

[2] Richard S. Sutton and Andrew G. Barto, *Reinforcement Learning: An Introduction*, MIT Press, 1998.

[3] Marilyn A. Walker, "An application of reinforcement learning to dialogue strategy selection in a spoken dialogue system for email," *Journal of Artificial Intelligence Research 12*, pp. 387–416, 2000.

[4] Esther Levin, Roberto Pieraccini, and Wieiand Eckert, "Using markov decision process for learning dialogue strategies," in *ICASSP*, 1998, pp. 201–204.

[5] Andrew G. Barto, Steven J. Bradtke, and Satinder P. Singhand, "Learning to act using real-time dynamic programming," *Artificial Intelligence Journal*, vol. 72, pp. 81–138, 1995.

[6] G. W. Furnas, S. Deerwester, S. T. Dumais, T. K. Landauer, R. A. Harshman, L. A. Streeter, and K. E. Lochbaum, "Information retrieval using a singular value decomposition model of latent semantic structure," in *SIGIR*. 1988, pp. 465–480, ACM Press.

[7] Thomas Hofmann, "Probabilistic latent semantic indexing," in *SIGIR*, 1999, pp. 50–57.

[8] Y.-C. Pan, Y.-Y. Liu, and L.-S. Lee, "Named entity recognition from spoken documents using global evidences and external knowledge sources with applications on mandarin chinese," in *ASRU*, 2005, pp. 296–301.

[9] S. E. Robertson, "On term selection for query expansion," *Jounal of Documentation*, vol. 46, pp. 129–146, 1990.