Interactive Spoken Document Retrieval With Suggested Key Terms Ranked by a Markov Decision Process

Yi-Cheng Pan, Hung-Yi Lee, and Lin-Shan Lee, *Fellow, IEEE*

***Abstract*—Interaction with users is a powerful strategy that potentially yields better information retrieval for all types of media, including text, images, and videos. While spoken document retrieval (SDR) is a crucial technology for multimedia access in the network era, it is also more challenging than text information retrieval because of the inevitable recognition errors. It is therefore reasonable to consider interactive functionalities for SDR systems. We propose an interactive SDR approach in which given the user's query, the system returns not only the retrieval results but also a short list of key terms describing distinct topics. The user selects these key terms to expand the query if the retrieval results are not satisfactory. The entire retrieval process is organized around a hierarchy of key terms that define the allowable state transitions; this is modeled by a Markov decision process, which is popularly used in spoken dialogue systems. By reinforcement learning with simulated users, the key terms on the short list are properly ranked such that the retrieval success rate is maximized while the number of interactive steps is minimized. Significant improvements over existing approaches were observed in preliminary experiments performed on information needs provided by real users. A prototype system was also implemented.**

***Index Terms*—Spoken document retrieval (SDR), dialogue system.**

I. Introduction

In THE future, the most attractive form of network content will no doubt be multimedia, which includes speech information. As the core concepts for the content can usually be found in this information, documents that contain speech—that is, spoken documents—will likely be the key for the retrieval and browsing of such content [1], [2]. There are many good example applications along this direction, including browsing applications for course lectures and broadcast news [3], [4], retrieval of meeting records [5], voicemail [6], podcasts [7], YouTube videos [8], and various other audio content over the World Wide Web [9]. In recent years, substantial research effort has been made in spoken document retrieval (SDR), and many successful techniques and systems have been developed [3], [4], [8]–[10]. Spontaneous speech, which typically includes many out-of-vocabulary (OOV) words and is recorded in adverse environments, yields relatively poor recognition accuracy. Approaches for this task include carefully designed robust indexing features and retrieval models, including those based on properly chosen subword units for handling OOV words [11], [12] and lattice-based approaches that take into account multiple recognition hypotheses [13]. Various topic analysis and concept matching approaches have also been developed to solve the term mismatch between the query and the desired spoken documents [14]–[16]. However, interactive search of spoken documents is still in its nascent stages.

Interactive information retrieval (IIR) [17], in which the retrieval process is made more effective by enriching the interaction between system and user, has been popularly used for text, image, and video for several years now. Relevance feedback-based interaction is a commonly used technique for image retrieval, because of the wide gap between high level concepts and low level features in the images, as well as the subjectivity of human visual perception [18]. IntentSearch [19] is a successful example of an online interactive image retrieval system. For video retrieval, the MediaMill video search engine [20] allows users to interact with the system via keywords, video examples, or even concepts; NewsRoom [21] and the Open-Video project [22] are two other well-known interactive video retrieval systems. The "Dialogue Navigator for Kyoto City" system [23], [24] uses efficient dialogue management techniques to help users navigate across Wikipedia documents about Kyoto as well as information for tourists from the Kyoto city government.

IIR techniques can also be applied to SDR, for several reasons. First, the query entered by the user during retrieval is usually quite short, because the user tries to describe his information need in the quickest and easiest way. This leads to ambiguity and thus many irrelevant documents are in the list returned by the system. This applies to the retrieval of all types of information, be it text, image, video, or speech. However, this problem is much more serious in SDR because browsing audio content requires much more effort than text content. Unlike text content, it is not easy to show audio content on the screen: the user does not know the content of the audio until he actually listens to it. Second, the user usually does not know how to

Manuscript received September 21, 2010; revised February 28, 2011 and July 11, 2011; accepted July 11, 2011. Date of publication September 15, 2011; date of current version December 21, 2011. This work was supported in part by the National Science Council, Taiwan, under Contract NSC 97-2221-E-002-134-MY3 and in part by the Ministry of Education under Contract 99R80303. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Gokhan Tur.

Y.-C. Pan is with MediaTek, Inc., Hsinchu 30078, Taiwan (e-mail: tlkagkb93901106@gmail.com).

H.-Y. Lee and L.-S. Lee are with the Department of Electrical Engineering, National Taiwan University, Taipei 10617, Taiwan (e-mail: tlkagkb93901106@yahoo.com.tw; lslee@gate.sinica.edu.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TASL.2011.2163512

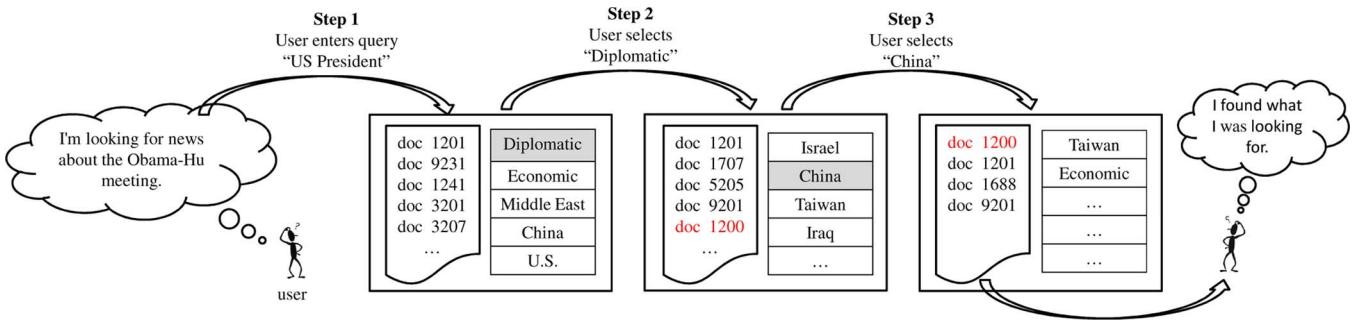

Fig. 1. Example of the system-user interaction when the user wishes to find news about "the meeting of Obama and Hu." The desired document is in red.

translate what he really needs in his mind into a good query, that is, one that efficiently yields the information he wants. Because the user usually does not know what can be found in the back-end archive and does not know how the retrieval system works, he does not know which query terms will retrieve the spoken documents he wants. This problem, too, is independent of the media type. Third, the OOV word problem in SDR is often handled using subword-based indexing and retrieval techniques, but such techniques inevitably lead to a significantly increased proportion of irrelevant retrieved documents and thus low precision. Follow-up interaction is therefore necessary for the user to identify and select the desired information.

In spoken language technology, recent decades have seen the wide application of many successful spoken dialogue systems with various application tasks and different capabilities [25]–[28]. Typical application tasks include travel information services, customer inquiry services, and car navigation. Such spoken dialogue technologies have yielded successful experiences for system-user interaction in spoken language systems. In most cases, the two major issues for these systems are spoken language understanding and dialogue modeling. Spoken language understanding is the transformation of the input speech utterance into a properly formed dialogue act with a parameter set: this can be readily mapped to a proper query for submission to a well-organized back-end database for the desired information. Dialogue modeling is directly related to system-user interaction, and maintains a fluent and efficient dialogue flow so that the requested task can be properly completed. Dialogue modeling approaches have progressed from rule-based methods to machine-learning approaches such as Markov decision processes (MDP) and partially observable Markov decision processes (POMDP) [28], [29]. Such machine learning approaches provide systematic methods to find interaction strategies that optimize objective functions related to the dialogue system performance. Many of the techniques used in these systems may be considered for interactive SDR.

We here propose an interactive SDR approach in which the system-user interaction is directed using a key term hierarchy. The suggested key terms are ranked using an MDP [30]–[33]. Preliminary experiments performed on Mandarin Chinese broadcast news yielded encouraging results, and a prototype interactive SDR system was implemented to demonstrate the concept. In Section II, we overview the approach, in Sections III and IV we describe in detail the technologies used, and in Section V we discuss the experimental results.

II. Overview of the Proposed Approach

We propose an interactive SDR approach in which after the user enters a query, the system returns not only the retrieval results but also a list of suggested key terms. From this list the user chooses a key term to expand his query and modify the retrieval results. This process can be repeated until the user obtains the documents he wishes to find, and is similar to existing term suggestion or interactive query expansion approaches [34] which have been well studied for interactive web search [35]. Most research assumes that the user is able to distinguish between good and bad terms, given a list of terms [36]. However, some have shown that it is difficult for users to make good interactive query expansion decisions [37]. We propose a novel approach in which we present in each interactive step properly ranked key term lists to help the user select a good key term for query expansion. The whole retrieval process is organized as a key term hierarchy and can be modeled by a Markov decision process (MDP), which we briefly summarize in this section.

A. Interactive SDR Scenario With Suggested Lists of Key Terms

Fig. 1 is a possible scenario of the proposed interactive SDR approach, and illustrates how the prototype system interacts with the user. The user is looking for news about the meeting of U.S. President Obama with Hu, the leader of China.[1] Since he does not know what can be found in the back-end archive, and does not know which query is more efficient, he enters the short query "US President." Although the term "US President" represents part of what he is looking for, he does not know that the query "US President" applies to many documents on different issues in the archive. In another case where the system accepts spoken query input, the user may submit the more precise spoken query "US President Obama and Hu." However, as the two name entities are both OOVs, only the words "US President" are correctly recognized; hence only the query "US President" is submitted to the system.

Such a short query can yield a large number of retrieved documents. Even if the news stories about the meeting of Obama with Hu are retrievable, they are inevitably buried among many other news stories about the US president, so most of the retrieved documents are not what the user wants. Because the user does not know what the audio is until he listens to it, it is difficult to find the desired story among the many retrieved

[1]This event took place in January 2011.

documents. Fortunately, besides the retrieval results, the system also returns a list of key terms for user interaction. As shown in Fig. 1, after submitting the initial query (step 1), listed in addition to the retrieval results is a list of key terms like "Diplomatic" and "Economic." In step 2, the user clicks on "Diplomatic" and the system refines the results accordingly. However, given the initial query "US President" and the key term "Diplomatic," the retrieval results include all of the U.S. diplomatic events, so it is still difficult to find the desired information. The refined key term list, though, includes terms like "Israel," "China," from which the user in step 3 selects "China." This process continues until the user is satisfied with the results. In this way, given the initial query and the follow-up key terms, the system can collect enough information to locate the documents the user wants. This, then, is system-user interaction using key term suggestion.

B. Ranking the Suggested Key Terms

One challenge for this interactive retrieval approach is how to assemble the set of key terms and then rank it properly based on the interaction history. This ranking is important for several reasons. First, as many users do not want to browse through the entire list, they simply select from the key term list the first term that looks helpful. Thus the term that they select depends heavily on the order of the terms. Second, even if the users are patient enough to browse the entire list, there are often several terms that can seem equally relevant. When users are not sure how to decide among the terms, the system suggests that they select the highest-ranking one. Third, users increasingly access network resources using hand-held devices with small displays. In such cases, users might not select any key terms that do not fit within the screen because they want to avoid scrolling.

Thus the ranking strategy must balance between two mutually contradictory properties—higher term coverage (the user's selection of the key term allows the system to retrieve more of the documents that the user wants to see) and higher discriminative power (key term selection allows the system to filter out many of the documents that the user does *not* want to see)—by properly reflecting the statistics and relationships among the documents in the archive. For example, in Fig. 1, because the selection of "U.S." in step 2 after the query "US President" does not give the system any extra information, the system should place the term lower in the list. That is, although the term "U.S." applies to most of the desired documents, it is not discriminative given the query "US President." On the other hand, say user A is looking for news about the Obama–Hu meeting, but user B wants news about all of Obama's diplomatic events. Given the query "US President," the key term "China" aids user A but may filter out many of the documents that user B wants. Hence, the key term "China" does not have enough coverage for user B. Since some users are like user A and some are like user B, a potential ranking strategy is to satisfy the needs as many users as possible. There could also be other considerations, like taking into account the speech recognition accuracy: key terms that are frequently misrecognized probably should be ranked lower.

C. MDP-Based Modeling of Interactive Spoken Document Retrieval

Markov decision processes (MDPs) have been very successful in modeling spoken dialogue systems. Here we use an MDP to model the interactive SDR process. Terminology for spoken dialogue systems such as dialogue session, action set, state space, and strategy all have analogue definitions in interactive information retrieval.

In the proposed approach, when a user wants some information, he submits a query to the system, which then returns the retrieval results and a list of suggested key terms. This is the system's action. When the user is not satisfied with the retrieval results, he selects a key term for query expansion, until he is satisfied, or he gives up. This is a *retrieval session*, which starts with the initial query and ends when the user either finds what he wants, or gives up.

A *reward* can be defined for each retrieval session. In spoken dialogue systems, a reward could be the number of interactions (dialogue turns) needed for successful completion of the desired task, or the task success rate. The reward for a retrieval session can be defined in similar ways: for example, it may have to do with number of interaction steps (submitting the query and selecting the key terms) the user takes before his information need is satisfied or the success rate for the retrieval sessions (the user does find what he wants). The goal here is to find the strategies that take the actions which optimize the reward.

The *action set* in interactive SDR is the set of all possible lists of key terms the system can return to the user. As mentioned in Section II-B, different key term rankings may lead to different retrieval results and hence different rewards for the retrieval session. Thus, different rankings are considered different actions.

The *state* of the interactive retrieval system is the interaction history. In the previous example, if the user submits "US President" and then selects "Diplomatic," the system is in the state [*US President, Diplomatic*]. On further selection of "China," the system moves to the state [*US President, Diplomatic, China*]. Thus, the initial state is determined by the initial query. At each step, the system returns a list of key terms for the user to select from. The choice of key terms to return to the user is based on a key term hierarchy constructed at the start of the retrieval session, based on the initial query; the key term hierarchy construction algorithm is described in Section III. The key terms in the list actually describe the transitions allowed between states. For example, in Fig. 1, after step 2, the transition from state [*US President, Diplomatic*] to states [*US President, Diplomatic, Israel*] and [*US President, Diplomatic, China*] are both allowed. When "Soccer" is not on the list after step 2, the transition from state [*US President, Diplomatic*] to state [*US President, Diplomatic, Soccer*] is not allowed. Corresponding to each state is also a set of retrieval results, which are shown to the user when the system is in that state. The final state is the state in which the user is satisfied by the corresponding retrieval results, or is not satisfied and gives up.

At each state, the proper action is simply the proper ranking of the returned key terms. This ranking is decided by the *strategy*. The core issue here is to find a strategy that optimizes the reward. In Section IV, we show that this can be achieved with an MDP

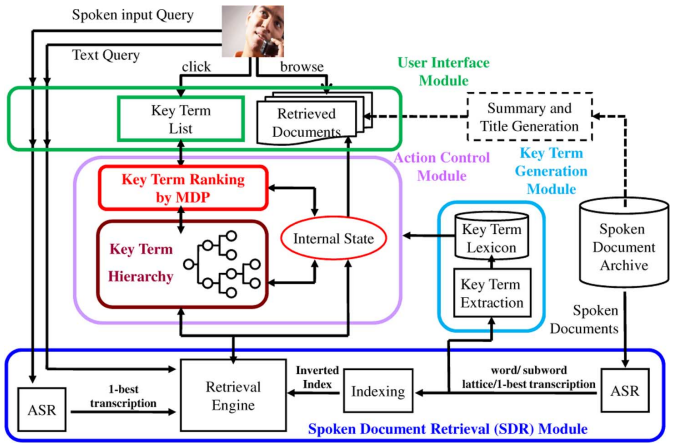

Fig. 2. Block diagram of the proposed interactive SDR approach.

trained with reinforcement learning [38] using a large number of simulated users, as has been done in spoken dialogue systems.

D. Block Diagram of the Proposed Approach

The block diagram for the proposed approach is shown in Fig. 2. The four main modules in the diagram include: 1) an SDR module (bottom) to search through the spoken document archive; 2) a key term generation module (middle right) for key term extraction and maintaining the key term lexicon; 3) an action control module (middle left) to choose the key terms for interaction, the key term rankings, and the spoken documents retrieved; and 4) a user interface module (top) for interaction.

The SDR module first transcribes the audio documents and constructs the search indices, which can be based on any type of transcription, including 1-best transcriptions or lattices with word or subword units. The search engine can be based on any kind of retrieval model, including a vector space model, latent semantic analysis (LSA), or probabilistic latent semantic analysis (PLSA). The user can submit either a text query or a spoken query to initiate the retrieval session; the latter is recognized by an ASR module and thereafter treated as a text query. The key term generation module then automatically extracts the whole set of key terms from the transcriptions of spoken documents for storage in a key term lexicon. Given a query, the retrieval engine produces an initial set of retrieved documents which the action control module then uses to construct a key term hierarchy. This hierarchy determines which key terms the module provides at each interactive step. The MDP is used to rank the key terms on the list shown to the user, and the internal state of the action control module decides which documents are retrieved. The user interface module presents the resultant document and key term lists for the user to browse through and select from, respectively.

E. Prototype System

We developed a prototype system using the proposed approach, in which Mandarin Chinese broadcast news stories were used as the spoken documents to be retrieved. In this system the news archive to be retrieved from and navigated across included roughly 110 hours of approximately 5800 news stories. All of the modules shown in Fig. 2 were implemented, including the extra "Summary and Title Generation" [39], [40] block in dotted

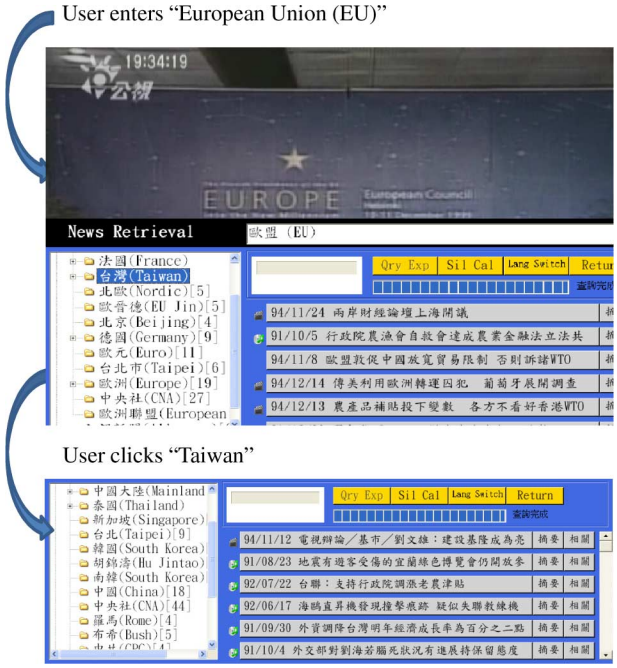

Fig. 3. Example screenshots of the prototype system. A user entered the initial query "European Union (EU)" first (upper part), and then he selected "Taiwan" for further interaction (lower part). The English version of the key terms were obtained via Google Translate.

lines on the top right corner. This extra block automatically generates a summary and a title for each news story in the archive to help the user more efficiently browse the retrieved documents. In this way, in addition to interacting with the system through the proposed approach, the user can also ask the system to show on the screen the automatically generated titles for the retrieved spoken documents, which he can browse and click on to listen to their summaries in speech form. Thus, the user can find the desired spoken documents without listening to every one of them, most of which are not what he is looking for. This enables the user to efficiently navigate the spoken document archives and obtain the desired information.

Fig. 3 shows some example screenshots of the prototype system. The upper part of Fig. 3 is the screenshot after entering the initial query "European Union (EU)." Listed in the right lower corner are the titles of the top several retrieved spoken documents; the most relevant news story is being played in the background. With the query "European Union," the suggested key terms are "France," "Taiwan," "Germany," and so on[2] which are the terms listed in the left lower corner. The user then selected "Taiwan" for further interaction. The screenshot is the lower part of Fig. 3. Selecting "Taiwan" brings up "Taipei" and "China" and so on for further selection.

III. Construction of the Key Term Hierarchy

We automatically extract as key terms a set of topically clearer and more representative terms to be used for system-user interaction. The number of key terms extracted from the archive can be very large. For better interaction, it is better to give the user a short list of key terms to choose from. Given the interaction history, not all the key terms are useful for interaction.

[2]All key terms were translated into English using Google Translate.

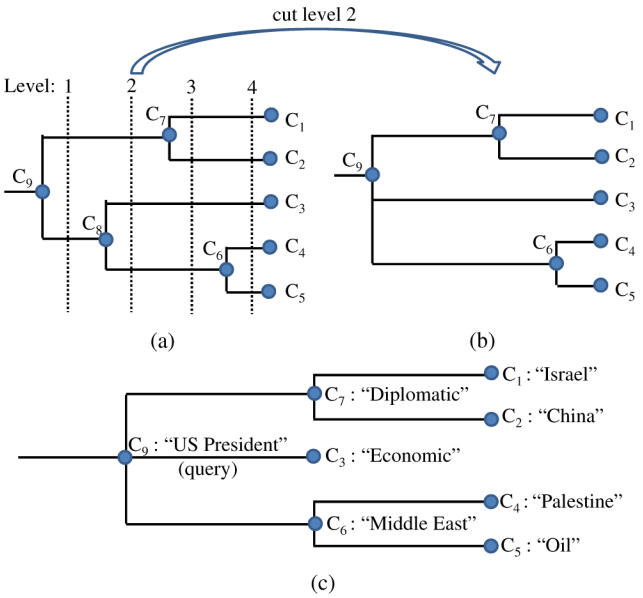

Fig. 4. Key term hierarchy construction. (a) Binary tree constructed using HAC. (b) m-ary hierarchy after the partitioning algorithm. (c) Part of the corresponding key term hierarchy from Fig. 1.

For example, if the user submits the query "US President," he is most likely not interested in "Software," and he would not select it even if it were an important key term. Hence, the key term list should include only terms that are highly related to the initial query. Moreover, given the query "US President," as the key terms "White House" and "U.S." may lead to similar sets of retrieved documents, they should not appear together on the list. Thus we see that what is shown to the user at each state should be a short list of the key terms that represent distinct "topics" that are highly related to the retrieval history; this is accomplished by organizing the whole retrieval session according to a key term hierarchy [30]–[33]. Fig. 4(c) is part of the hierarchy around which the retrieval session in Fig. 1 is structured. Every node on the hierarchy is a key term, and the root is the initial query ("US President"). After entering the initial query, at steps 2 ("Diplomatic") and 3 ("China") the user traverses down the hierarchy by selecting a key term at the second and third levels of the hierarchy.

Here a key term hierarchy is constructed by clustering the documents retrieved by the initial query, so the key term hierarchy (that is, the allowed state transitions) is decided at the beginning of a retrieval session. Consider a user who submits the query "US President" but is not satisfied with the results. If we cluster the documents retrieved by the query "US President," we find that there are several major clusters in the retrieved documents, one for documents including "Diplomatic," another for "Economic," another for "Middle East," and so on. The documents that the user wants likely belong to one of these clusters. To identify which is the right cluster, just the terms "Diplomatic," "Economic," and "Middle East" may suffice. As further analysis of the "Diplomatic" cluster may yield sub-clusters for "Israel," "China," and so on, the key terms "Israel" and "China" may describe what the user wants after submitting the query and clicking on "Diplomatic." This is why a hierarchy constructed by clustering the retrieved documents can help the system find a short key term list with relatively distinct topics given the interaction history. Such a hierarchy can be constructed using a clustering algorithm.

To construct the key term hierarchy for interaction, first a general key term set is extracted from the spoken archive offline based on the technique described in Section III-A. After the user enters the initial query, the key term hierarchy is constructed using the hierarchical agglomerative clustering and partitioning algorithm $(\mathrm{HAC}+\mathrm{P})$ [41] which includes the hierarchical agglomerative clustering (HAC) algorithm and the partitioning (P) algorithm. The procedure of $\mathrm{HAC}+\mathrm{P}$ is summarized as follows:

1) initialization of the clusters;
2) use of the HAC algorithm to construct a binary tree;
3) use of the partition algorithm to transform the binary tree into a balanced and comprehensive m-ary hierarchy, where m can be different integers at different splitting nodes.

These steps are presented respectively in Sections III-B–III-D. After using this algorithm to construct the hierarchical clustering of documents, we construct the key term hierarchy based on the node labeling of the hierarchy structure in Section III-E. Experiments on the efficiency of the constructed key term hierarchy are presented in Section V-D.

A. Automatic Key Term Extraction

Many approaches for automatically extracting key terms have been proposed [42], [43]. Compared to features such as training corpus occurrence counts, inverse document frequencies and so on, it was found that latent topic entropy is especially useful in extracting key terms, at least for the corpora used in the experiments reported here [15]. We thus use latent topic entropy as the key term selection feature. The latent topic entropy of term t is based on PLSA [44] and is defined as

$$E_{\mathrm{top}}(t) = -\sum_{k=1}^{K} P(z_k \,|\, t) \log P(z_k \,|\, t) \tag{1}$$

where z_k is a PLSA latent topic, K is the total number of latent topics, and $P(z_k \,|\, t)$ is the probability that topic z_k is being addressed given term t. A lower $E_{\mathrm{top}}(t)$ indicates that t, because it is concentrated on fewer latent topics and carries more topical information, may be a possible key term. Here, terms with a latent topic entropy lower than a threshold are taken as key terms.

B. Cluster Initialization

Given the initial query, we first retrieve a document set and gather the key terms appearing in these documents, with which we define the initial clusters as all documents that include the same key term t. Because a document may include more than one key term, that document may belong to more than one initial cluster; this is reasonable because a document may be semantically related to more than one topic. Note that here we use key term-based initial document clusters instead of clustering the documents directly, because key terms are usually semantically clearer, more identifiable, and discriminative than the documents themselves. For each key term t, each of which represents an initial cluster, we construct feature vector v_t by averaging the vector representations v_d of all the documents that

include the term, weighted by the term's frequencies in the documents. Here words and subword units and their n-grams can be used to represent document d as vector v_d. HAC + P can then be performed on these feature vectors v_t to cluster the initial clusters into a balanced m-ary hierarchy.

C. HAC Algorithm

The HAC algorithm is based on the similarity defined between clusters C_i and C_j, $S(C_i, C_j)$:

$$S(C_i, C_j) = \frac{1}{|C_i|\,|C_j|} \sum_{v_t \in C_i} \sum_{v_s \in C_j} c(v_t, v_s) \tag{2}$$

where $c(v_t, v_s)$ is here simply the cosine measure of the angle between vectors v_t and v_s for key terms t and s, and $|C_i|$ is the total number of documents in C_i. Thus, $S(C_i, C_j)$ is the averaged cosine similarity between each pair of v_t in C_i and v_s in C_j. The HAC algorithm is performed bottom-up from the initial clusters. Assume there are n initial clusters, $C_1, C_2, \ldots, C_n$, one per key term, the elements of which are the documents in which the corresponding key term occur. At each step, two clusters with the largest similarity defined in (2) are merged. Let C_{n+i}, $i = 1, 2, \ldots, n-1$, be the new cluster created at the ith step by merging the two clusters. The output binary tree can thus be expressed as a list, $C_1, \ldots, C_n, C_{n+1}, \ldots, C_{2n-1}$, where $C_{n+1}, \ldots, C_{2n-1}$ are the new clusters created by HAC. An example is in Fig. 4(a), where $n = 5$ and $C_6, \ldots, C_9$ are created by HAC.

D. Partitioning Algorithm

The next phase, the partitioning (P) algorithm, is performed top-down. The binary tree is partitioned by a horizontal cut first into several sub-hierarchies, and then this procedure is repeated recursively on each sub-hierarchy. In each such partitioning procedure the best level at which the binary tree should be cut is selected to yield the best set of sub-hierarchies. Here the best set of sub-hierarchies is determined based on the balance of two parameters: *cluster set quality* and *number preference score*, as will be explained below. In Fig. 4(a), the partition can be performed by a cut through the binary tree on four possible levels: $l = 1, 2, 3$, and 4. For a cut performed at the $l = 2$, the result is the three sub-hierarchies C_3, C_6, and C_7 in Fig. 4(b), where cluster C_8 is deleted and the splitting node C_9 has three branches $(m = 3)$.

Cluster set quality is based on the concept that each cluster should be as cohesive and isolated from other clusters as possible. Therefore, the cluster set quality of a cluster set H is defined here as

$$Q(H) = \frac{1}{|H|} \sum_{C_i \in H} \frac{S(C_i, \overline{C_i})}{S(C_i, C_i)} \tag{3}$$

where C_i is a cluster for a sub-hierarchy produced after the partitioning cut, $\overline{C_i} = \bigcup_{k \neq i} C_k$ is the complement of C_i, $S(C_i, C_j)$ is defined as in (2), and $|H|$ is the number of clusters in H. The smaller the value of $Q(H)$ the better the set quality.

In principle, when a hierarchy node is split into m sub-hierarchies, m should be neither too small nor too large, taking into consideration both the efficiency (not too small) and the convenience (not too large) for the user. In this algorithm, the preferred number m_0 is defined as a design parameter to be assigned when constructing the hierarchy. A gamma distribution function is used as an approximated score to measure how close the obtained value of m is to the preferred number m_0:

$$f(m) = \frac{1}{\alpha! \beta^{\alpha}} m^{\alpha - 1} e^{-m/\beta} \tag{4}$$

where m is the number of sub-hierarchies at the splitting node obtained using the partitioning algorithm, α is a positive integer, and the constraint $(\alpha - 1)\beta = m_0$ gives $f(m_0) \geq f(m)$, that is, $f(m)$ is maximized when m is equal to the preferred number m_0 [45]. Here we set m_0 to be the largest integer smaller than the square root of the number of leaf nodes for the partitioning being considered.

With the two parameters $Q(H)$ and $f(m)$ defined in (3) and (4), the best level of partitioning cut is then chosen as the one which minimizes

$$\eta = Q(H)/f(m) \tag{5}$$

with which $Q(H)$ is minimized while $f(m)$ is maximized. In this way, we partition the previously constructed binary tree step-by-step, in a top-down fashion, to construct a balanced and comprehensive m-ary hierarchy.

E. Hierarchical Document Clustering-Based Key Term Hierarchy

After the hierarchical clustering of documents is constructed, each cluster on the hierarchy is labeled with a key term; this node labeling defines the key term hierarchy used to structure the retrieval session. The root of the hierarchy is labeled by the initial query q. Assume Fig. 4 is the hierarchy structure for the documents retrieved for query "US President" in Fig. 1. Root node C_9 is labeled as "US President," and the other nodes are labeled with the most frequently occurring key terms in the clusters which have not been used as labels by their ancestors. For example, the most frequent key term in C_7 may well be "US President," but because this term has been used already by C_9, C_7 is labeled with "Diplomatic," the cluster's second-most frequent key term. Sibling nodes are merged if they are labeled with the same key term.

IV. MDP-Based Key Term Ranking

In this section, we describe how to rank the key terms in the list shown to the user at each state so as to optimize the interaction, for example by minimizing the number of steps a user must take before he finds what he wants, or by maximizing the retrieval session success rate.

A. State

Here each state s of the system is represented by the initial query and the following key terms. At each state s the retrieved document set $G(s)$ is shown to the user. Assume the user is looking for something and starts with query q: the allowed state transitions are first determined using the procedures in Section III, and then all of the possible state paths for the session are listed and structured as a state path tree like that in Fig. 5. Each tree node is a state, and the arrows connecting the

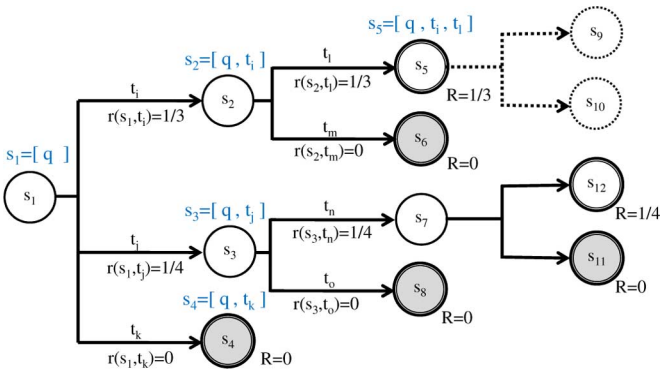

Fig. 5. State path tree for the retrieval sessions for a given user.

states represent the possible state transitions corresponding to the key terms which can be selected at the states. A retrieval session is a state path starting from the root (s_1, the initial state) and ending at a leaf (final state), that is, a node with a double circle.

In Fig. 5, the user initiates the retrieval session by submitting query q: this is denoted as state s_1 at the root, that is, $s_1 = [q]$. s_1 maps to the set of documents $G(s_1)$ returned when the query is submitted directly to the search engine in the SDR module. Assume the user is not satisfied with the retrieval results $G(s_1)$ for this state, so the system provides key terms t_i, t_j, and t_k, which the user can select to more clearly describe what he wants. With this selection, the system transits to one of the states s_2, s_3, or s_4:

$$s_2 = [q, t_i], \quad s_3 = [q, t_j], \quad s_4 = [q, t_k].$$

If the user then selects key term t_i, the system moves to state $s_2 = [q, t_i]$. The search engine starts a new retrieval process using t_i as the query to search through the whole archive, and the retrieval results $G(s_2)$ at state s_2 are the intersection of the two retrieved document sets for queries q and t_i. This process continues. For example, if the retrieval results $G(s_2)$ for state s_2 are not what the user wants, key terms t_l or t_m can be further selected. If the user selects t_l, the system goes to state $s_5 = [q, t_i, t_l]$, and so on.

The key term hierarchy which determines the state transitions constrains the state path tree, but not all the states which can be visited according to the key term hierarchy are included in the state path tree. For example, in Fig. 5, the state s_5 can further transit to s_9 or s_{10}, but the user leaves the system at s_5 because he is already satisfied with the retrieval results $G(s_5)$. Therefore, s_9 and s_{10} are not included in the state path tree.

B. Reward

We mentioned in Section II-C that a core issue of the interactive SDR approach is to find the strategy which optimizes a pre-defined reward for the retrieval session. We here define the reward R as

$$R = \begin{cases} 1/n, & \text{if the retrieval session is successful} \\ & \quad \text{(or the user is satisfied)} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

where n is the number of steps the user takes in interacting with the system before he is satisfied, including submission of the initial query and all following key term selections. A retrieval session is defined to be successful (that is, the user is defined to be satisfied) if at the end of the session (at the final state) a certain selected information retrieval (IR) metric (for example, F-measure) of the retrieved documents is above a pre-defined threshold θ. The highest possible reward for a retrieval session is 1: this occurs when the metric of the retrieval results of the initial query already exceeds the threshold. The reward is lower if more steps are needed for the user to be satisfied; it is reduced to zero if the retrieval session is a failure, that is, the IR metric is still below the threshold when the system state corresponds to a leaf node of the key term hierarchy, when no more key terms can be selected. The larger the reward R the better.

In Fig. 5, state paths that end at white leaf nodes indicate the retrieval session is successful, while paths that end at grey leaf node are failures. A session with the state path s_1, s_2, and s_5 is successful if the retrieved document set at state $s_5 = [q, t_i, t_l]$ has an IR metric above the threshold. The session reward for that state path is then 1/3 based on (6) because the user finished in three steps. On the other hand, a session with the path s_1, s_2, and s_6 is a failure if the retrieved results $G(s_6)$ at the state $s_6 = [q, t_i, t_m]$ are not satisfactory to the user, and in addition there is no further key term which can be selected at s_6. The session reward for this state path is thus 0.

C. Reward Optimization

The role of MDP is to guide the retrieval sessions along state paths that yield the maximum rewards. At each state, the selection of different key terms leads to different state paths with different rewards. Consider Fig. 5: at s_1, the highest possible reward is 1/3 if the user selects t_i and then t_l to reach s_2 and s_5. However, if he instead selects t_j at s_1 to get to s_3, he has already lost the chance to reach s_5. In this case, the best reachable final state from s_3 is s_{12} with a reward of only 1/4. Selecting the term t_k will only lead him to a reward of 0. When selecting key term t at the state s, we denote the maximum potential reward as $r(s, t)$. Thus,

$$r(s_1, t_i) = 1/3, r(s_1, t_j) = 1/4, \text{ and } r(s_1, t_k) = 0.$$

For this example t_i is the best choice at s_1. That is, if we can look ahead to the end of the retrieval session, at state s_1 we know the best choice among the key terms t_i, t_k, t_j is t_i. Assuming that the user browses the key term list from the top and selects the first key term he is interested in, clearly we should rank t_i above t_j, and t_j above t_k at state s_1. More generally, we should rank key term t at state s based on the values of $r(s, t)$ for all different t in a decreasing order. However, note that although the user knows what he is looking for, he does not know what can be retrieved from the archive, so has no idea what reward he can expect when he selects a key term. This leaves us with the task of estimating the value of $r(s, t)$ for all key terms t at each state s. Note that because the system does have the whole archive, it knows what can be retrieved given any key term selection: what it does not know is what the user wants. We address this in the next two sections.

D. Reinforcement Learning

Although we do not know in advance what each specific user wants, we can train the system to work well for most users, if we have enough training data, that is, the information needs and the initial queries of many users[3]. This is because the expected maximum obtainable reward at the end of a retrieval session given the selection of key term t at state s can be estimated with the training data. In the example in Section II-B, if the majority of users who entered the query "US President" were looking for all US diplomatic events (user B in Section II-B), and only a minority were looking for a single event (user A), it would be prudent to rank "Diplomatic" higher than "China" given the initial query "US President." Reinforcement learning using a large corpus with many training users can pick up on this knowledge and improve the system for most users.

Assume the training corpus includes a large number of data sets, in which each set includes the desired document set (or information need) D of a training user and the initial query q entered to find D. For each set, the system first uses the query to obtain from the archive the retrieved documents, which it uses to construct a key term hierarchy, which it then uses to layout the complete state path tree such as that in Fig. 5. The system then computes the maximum reward obtainable, $r(s,t)$, for each key term selection for each state on the state path tree for this training user. We then estimate the expected value of $r(s,t)$, $E[r(s,t)]$, by averaging over all of the training users. The complete algorithm is summarized in Algorithm 1, where $Q(s,t)$ is the accumulated value of $r(s,t)$, and $n(s,t)$ is its accumulated count.

Algorithm 1 Estimate $E[r(s,t)]$, the expected value of $r(s,t)$ for every key term t at each state s

Input:

a set of training users $\{D_1, q_1\}, \{D_2, q_2\}, \ldots, \{D_M, q_M\}$

Output: $E[r(s,t)]$

$Q(s,t) = 0$

$n(s,t) = 0$

for each training user $\{D_i, q_i\}$ **do**

  Construct a state path tree based on $\{D_i, q_i\}$

  for every state s on the state path tree **do**

    for every key term t selectable at state s **do**

      $n(s,t) = n(s,t) + 1$

      $Q(s,t) = Q(s,t) + r(s,t)$

    end for

  end for

end for

$E[r(s,t)] = Q(s,t)/n(s,t)$

[3]For an online search engine, this training data can be gleaned from user logs [46].

When a query is submitted by a test user, although the system does not know what the test user wants, at each state s all of the listed key terms t are ranked according to $E[r(s,t)]$.

E. Simulated Users

For the reinforcement learning algorithm, we need a large corpus of training users. When such a corpus is not available, a simulated corpus with similar properties may be automatically generated for the purpose with a huge number of simulated users. Although the simulated users perhaps do not behave like real users, a system trained by simulated users can still be tested and the performance evaluated. We take this approach here. Below we describe how we generate the simulated users.

Each simulated user is represented by training data containing the desired document set D and the initial query q. In observing data for real users, we found that the document sets shared two characteristics: 1) most of the documents d in a given document set D had similar PLSA topic distributions $\{P(z_k \mid d), k = 1, 2, \ldots, K\}$, where z_k is a latent PLSA topic and K the total number of topics; and 2) most documents in the document set D usually shared common key terms. Based on these observations, we thus generate simulated users, each of which is represented by a desired document set D and an initial query q.

1) All of the spoken documents in the archive are first clustered into N_0 clusters using the PLSA topic distribution of each document and the k-means algorithm.
2) Cluster c is randomly chosen from N_0 clusters. The probability that a specific cluster is chosen is proportional to the size of that cluster.
3) Key term t is randomly chosen from key term set $\mathcal{K}$ (the key term lexicon). The probability that a specific key term is chosen is proportional to the number of documents in c in which the key term occurs. Hence, key terms that are never observed in c are not chosen.
4) Integer M is randomly generated as the size of the desired document set D.
5) The desired document set D is generated using Algorithm 2. This algorithm selects the key term t' in $\mathcal{K}$ most semantically related to t (in the sense of cosine similarity between the PLSA topic distributions), and adds to the document set $\hat{D}$ those documents d in c in which t' occurs. This procedure is repeated (the first t' is t itself, and the second is the next key term closest to t, and so on) until the size of $\hat{D}$ exceeds the desired size M. D is then the set of M documents randomly sampled from $\hat{D}$.
6) Randomly select as the initial query a key term q that occurs in at least one document in D.

With the above steps, we can generate any number of simulated users needed for reinforcement learning.

Algorithm 2 Generate the desired document set D for a given simulated user

Input:

a document cluster c, key term t, key term set $\mathcal{K}$, an integer M

Output: D

$\hat{D} = \phi$

while the size of $\hat{D}$ is smaller than M **do**

$t' =$ the key term in $\mathcal{K}$ closest to t based on PLSA

for each d in which t' occurs **do**

if $d \in c$ **then**

$\hat{D} = \hat{D} \cup \{d\}$

end if

end for

delete t' from $\mathcal{K}$

end while

D is the set of randomly sampled M documents from $\hat{D}$

V. Experimental Results

A. Experimental Setup

In the experiments, we used broadcast news stories in Mandarin Chinese as the spoken document archive to be retrieved from and navigated across. All of the news stories were recorded from radio or TV stations in Taipei from 2001 to 2003. There were a total of 5047 news stories, with a total length of 96 hours. The news stories ranged in length from 68 to 2934 characters with an average of 411 characters per story. 185 sets of real retrieval data were provided by 22 graduate students, each of which included the desired document set D and initial query q, where q had both a text version and a voice version. The number of desired documents in D ranged from 1 to 50 with an average of 19.5, and the query length ranged from 1 to 4 Chinese words with an average of 1.6 words, or 1 to 8 Chinese characters with an average of 2.7 characters. 22 sets out of the 185 were used for parameter tuning, and the other 163 sets were used as 163 testing retrieval sessions, or 163 test users. These 163 testing retrieval sessions were all generated by real users, and were used to test the approach trained using simulated users.

For recognition we used a 60 K-word lexicon and a tri-gram language model trained on 39M words of Yahoo news. We used different acoustic models for transcribing the spoken documents and spoken queries in order to conduct evaluations for different recognition accuracies. As listed below, we used four different recognition conditions for the spoken documents.

- doc (A): Manual transcription (100% accurate).
- doc (B): The spoken documents were recognized by a set of acoustic models with 64 Gaussian mixtures per state trained on a corpus of 24.5 hours of broadcast news different from the archive tested here. The character accuracy for the speech archive was 61.44%.
- doc (C): Same as doc (B), but with eight Gaussian mixtures per state. The character accuracy for the speech archive was 52.88%.
- doc (D): The acoustic models were trained on 24.6 hours of data highly mismatched to the target archive, but with 24 Gaussian mixtures per state. The character accuracy of the speech archive was 49.85%.

The queries were also recognized under different recognition conditions.

- qry (1): Text queries (100% accurate).
- qry (2): The spoken queries were recognized by the same set of acoustic models used in doc (D), but adapted using class-based MLLR based on 100 utterances of the speaker that produced each spoken query. The character accuracy of the 163 spoken queries was 80.80%.
- qry (3): Same as qry (2) but without MLLR, with a character accuracy of 61.64%.

A 64-topic PLSA model was trained using the 1-best output of the four different versions of document archive doc (A), (B), (C), and (D) respectively for each case to be used in key term extraction, and simulated user generation.

Any IR metrics can be used for the definition of a successful session, or user satisfaction, as mentioned in Section IV-B, and here we used F-measure, and set the threshold θ to 0.2. Hence, a retrieval session was considered successful at a certain state if the F-measure of the retrieval results corresponding to that state exceeded 0.2, and that state was taken as the final state. A session was a failure if there were no more key terms to select (already at a leaf of the key term hierarchy), but the F-measure at that state was still below 0.2. F-measure is a suitable IR metric for the task considered here. This is because for the case here the user can have a returned document set containing many relevant documents and excluding irrelevant documents by the key term interaction, so he does not have to scroll the screen and browse the returned list to find what he wants; thus, the ranking performance is not crucial here. Therefore, IR metrics considering ranking such as mean average precision (MAP) are not considered here.

B. Non-Interactive Retrieval Results for Initial Queries

In the following experiments, for retrieval (the SDR module of Fig. 2) we integrated position specific posterior lattices (PSPLs) constructed with words, Chinese characters, and Mandarin syllables. The n-gram matching scores obtained from the three PSPLs constructed using different units were weighted and summed as the relevance score. Given a query, we retrieved the documents whose relevance scores were higher than a threshold. The first experiment was direct retrieval, that is, using the initial queries without any interaction. For the four types of document archives, doc (A), (B), (C), and (D), we constructed lattices using the same beamwidth so the lattices and PSPLs would have the same size (but with different quality for the four different cases). These four different document archives and the three sets of initial queries yielded twelve different sets of retrieval results, all with different accuracies. The precision, recall and F-measure of these retrieval result sets are listed in Table I. The recalls were all relatively high, but the precisions were all quite low, so the F-measures were all very low. This was the actual scenario for the example in Section II-A. With the short initial queries, many documents were retrieved, but most of them were not what the user wanted. Table I also shows that performance in general degraded with lower recognition accuracies, and that recognition errors in the queries had a greater impact on performance than recognition errors in the documents. This suggests that the following

TABLE I
PRECISION, RECALL, AND F-MEASURE OF NON-INTERACTIVE, DIRECT RETRIEVAL

		Query accuracy								
		qry (1)			qry (2)			qry (3)		
		F-measure	Precision	Recall	F-measure	Precision	Recall	F-measure	Precision	Recall
Document Accuracy	doc (A)	8.13%	4.24%	99.94%	4.75%	2.47%	62.77%	3.08%	1.60%	40.29%
	doc (B)	6.36%	3.30%	86.92%	4.45%	2.31%	59.23%	3.08%	1.60%	41.08%
	doc (C)	6.19%	3.21%	82.60%	4.23%	2.20%	56.46%	3.00%	1.56%	40.38%
	doc (d)	5.82%	3.02%	79.35%	4.23%	2.20%	56.40%	2.91%	1.51%	39.81%

TABLE II
NUMBER OF KEY TERMS EXTRACTED UNDER DIFFERENT RECOGNITION CONDITIONS

	Document accuracy			
	doc (A)	doc (B)	doc (C)	doc (D)
Key term count	7472	9279	9465	9203

measures to introduce system-user interaction may yield better results for users.

C. Key Term Extraction

For key term extraction we used the latent topic entropy from (1) in Section III-A. We selected as key terms each word in the transcriptions with a latent topic entropy lower than 0.5 and a term frequency (counted with the 1-best results for the four different versions of document archive with different recognition accuracies) between 10 to 100. This means that different recognition accuracies yielded different sets of key terms for the same speech archive. Table II lists the number of key terms for the different recognition conditions.

D. Key Term Hierarchy-Defined State Transitions

As presented in Section III, the allowed state transitions are defined by the key term hierarchy which is constructed from the documents retrieved for the initial query. Therefore, we sought to determine whether the possible state paths constrained in this way really lead to successful sessions. If this key term hierarchy does not yield some successful state paths in a retrieval session, the retrieval session will never be successful, regardless of the key terms selected at each state. In addition, in Section III-B, the $\mathrm{HAC}+\mathrm{P}$ algorithm for constructing the hierarchy is initialized from the vector representations v_t for the key terms, which is obtained by averaging many document vector representations v_d. We wanted to identify the best kinds of vector representations v_d for such purposes. For different vector representations v_d used in the $\mathrm{HAC}+\mathrm{P}$ algorithm, we computed the percentage of the 163 tested retrieval sessions for which the state path tree constrained by the key term hierarchies did include at least one state path leading to successful retrieval sessions. Here, we used the tri-gram counts for words, characters, and syllables, all weighted by inverse document frequency, as the different document features used in the vector representation v_d.

The results are listed in Table III. We can see that in the case of manual transcriptions [doc (A)], all of the 163 total retrieval sessions had at least one successful state path, regardless of the choice of vector representations or the recognition accuracy of spoken queries. Even for the worst accuracies—doc (D) plus qry (3)—84.7% to 85.9% of the sessions had at least one successful path defined in the key term hierarchy. This shows that the state transitions defined by the key term hierarchies are actually quite reasonable. Table III also shows that syllable tri-gram counts worked slightly better than character tri-gram counts, which were slightly better than word tri-gram counts. This is reasonable because in Mandarin recognition, syllable accuracy is usually the highest and word accuracy the lowest. That is, it is relatively simple to correctly recognize syllables, but OOV words and word segmentation problems make correctly decoding the syllables into words much more difficult. For this reason, in the following experiments we used as the feature for document representation in the $\mathrm{HAC}+\mathrm{P}$ algorithm syllable tri-gram counts weighted by inverse document frequency. However, take the condition of doc (B) plus qry (1) with syllable tri-gram counts as an example: in average there were 5626 possible state paths in a retrieval session, but only 22 of them led to success retrieval sessions (about 0.4%), although 95.1% of the retrieval sessions had successful state paths. This shows that it is important to direct the system along a correct state path which minimizes the interaction needed for a successful session.

E. Number of Simulated Users

We also need to know how many simulated users are necessary to train a good key term ranking strategy. For the simulated user generation from Section IV-E, the documents were clustered into 16 clusters $(N_0 = 16)$, and the size of the desire document set of simulated user M was an integer uniformly distributed between [1,50]. We trained with user counts from 1 000 to 500 000 for the doc (B) plus qry (1) condition. Listed in Table IV are the results for $\bar{R}$, the reward R averaged over the 163 sessions. As we found that the averaged reward saturated when the number of simulated users was 100 000, we used this number for the simulated user experiments that follow.

F. Key Term Ranking

Again, we used reward R from (6) to evaluate the achievement of each retrieval session. In this section, we compare the performance of different key term ranking strategies using this reward function averaged over the 163 retrieval sessions tested. We also calculated the task success rate r_{success} over the 163 sessions and the average number of interaction steps n_{step} needed for those successful sessions. Note that the F-measure obtained with direct retrieval without interaction in Table I ranged from 0.029 to 0.081 for different recognition conditions; that is, without interaction, most retrieval sessions were poor. Because we here set the F-measure as the IR metric used for the definition of a successful session, and the threshold θ is 0.2. the goal here is to try to raise the F-measure to 0.2 as much as possible using interaction.

TABLE III
RETRIEVAL SESSIONS WITH AT LEAST ONE SUCCESSFUL STATE PATH IN THE KEY TERM HIERARCHY-CONSTRAINED TREE, FOR DIFFERENT DOCUMENT REPRESENTATIONS IN THE HAC + P ALGORITHM

		Query Accuracy								
		qry (1)			qry (2)			qry (3)		
		Word tri-gram	Character tri-gram	Syllable tri-gram	Word tri-gram	Character tri-gram	Syllable tri-gram	Word tri-gram	Character tri-gram	Syllable tri-gram
Document Accuracy	doc (A)	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
	doc (B)	93.9%	94.6%	95.1%	93.8%	93.9%	93.9%	91.8%	92.9%	93.2%
	doc (C)	91.5%	94.2%	94.6%	90.0%	93.1%	93.5%	89.3%	92.5%	92.9%
	doc (D)	86.4%	87.4%	88.1%	86.0%	87.4%	87.4%	84.7%	85.9%	85.9%

We used the 163 retrieval sessions, each of which included a desired document set D and an initial query q provided by 22 graduate students. Before the user was satisfied, it was assumed that the user followed the ranking order of the key terms on the list offered by the key term ranking strategies: he first checked to see whether the key term on the top of list was relevant to his information need, that is, whether that key term would retrieve at least one of the documents in his desired document set D. If not, the user then moved to the next key term on the list, and so on. Once the first key term relevant to his information need was found, the user simply selected that key term as the next step, as described in Section II-A. This yielded another set of retrieved documents and another ranked key term list which were shown to the user. This process continued until the F-measure of the retrieved documents based on the desired document set D exceeded 0.2 (a successful session), or until there were no more key terms to select (a failure).

We compared the proposed approach with four other key term ranking strategies: *random, tfidf, wpq* [47], and local context analysis (*lca*) [48] as briefly explained below.

- *random*: key terms ranked randomly.
- *tfidf*: key term t ranked according to

$$\mathrm{tfidf}(t) = tf(t) \times \mathrm{idf}(t) \tag{7}$$

where $tf(t)$ is the term frequency of t in the 1-best results of the whole spoken archive, and $idf(t)$ is the inverse document frequency of t.

- *wpq*: key term t ranked according to

$$wpq(t) = \left[\frac{r_t}{L} - \frac{n_t - r_t}{N - L}\right] \times \log \frac{r_t/L - r_t}{(n_t - r_t)/(N - n_t - L + r_t)} \tag{8}$$

where r_t is the number of relevant documents in which t occurs, n_t is the number of all documents in which t occurs, L is the total number of relevant documents, and N is the total number of documents. Since the system never knows which documents are relevant, the m documents with the highest relevance score for the query were considered relevant [49].

- *lca*: assumes that key terms that frequently co-occur with the query in the same document are good key terms for selection. So key term t is ranked according to

$$\mathrm{lca}(t) = co(t, q) \times idf(t) \tag{9}$$

where $co(t, q)$ is the number of co-occurrences between t and q in all documents [48].

TABLE IV
AVERAGED REWARD $\bar{R}$ FOR VARIOUS NUMBERS OF SIMULATED USERS FOR DOC (B) PLUS QRY (A)

	1k	5k	10k	50k	100k	500K
$\bar{R}$	0.261	0.317	0.326	0.331	0.336	0.336

Listed in Table V are the task success rate r_{success}, the average number of interaction steps for successful sessions n_{step}, and the averaged reward $\bar{R}$ from (6) of Section IV-C for the proposed approach in comparison with *random, tfidf, wpq*, and *lca* under different recognition conditions. The columns are grouped by query recognition accuracy [qry (1), (2), and (3)], whereas the four horizontal sections are for the various spoken document recognition accuracies [doc (A), (B), (C) and (D)]. The paired t-test was used for significance testing in the following experiments; results were considered significantly different if $p < 0.05$. The standard deviation of the interaction steps for successful sessions is shown beside n_{step}. In this table the superscripts α, β, γ, and δ, respectively, indicate that the results are significantly better than those of the *random, tfidf, wpq*, or *lca* approaches. From the left section (r_{success}), we see that the proposed approach outperformed the other approaches, with the exception of *lca* in the doc (A) plus qry (1) case. We discuss this exception below. It is also clear that r_{success} degraded when the recognition accuracy degraded. From the middle section (n_{step}) as well, we see that the proposed approach performed best in all cases, although in some cases the difference was less significant. We discuss this also below. The results for $\bar{R}$ in the right section are similar: the proposed approach was significantly better in all cases. Note that the task success rate r_{success} and the average number of steps needed for successful session n_{step} are parameters that are directly perceived by the users, while the averaged reward $\bar{R}$ is an integrated parameter which reflects both r_{success} and n_{step}.

By maximizing $\bar{R}$ with reinforcement learning as described in Section IV, we simultaneously make r_{success} larger and n_{step} smaller. Hence, it is not surprising that the averaged reward $\bar{R}$ was better for the proposed approach, because it was optimized by the reinforcement training. The higher r_{success} and smaller n_{step} were good for the user.

In Table V, we see that the proposed approach significantly outperformed the other approaches in almost all cases. We discuss this in Section V.G. Here we note that both *wpq* and *lca* worked reasonably well when both documents and queries were text [doc (A) plus qry (1)], but this did not hold for other conditions with more recognition errors. This was probably because *wpq* and *lca* were developed for text information retrieval, for

TABLE V
r_{Success}, n_{Step}, AND $\bar{R}$ FOR THE PROPOSED APPROACH COMPARED WITH *RANDOM, TFIDF, WPQ*, AND *LCA* METHODS UNDER DIFFERENT RECOGNITION CONDITIONS. SUPERSCRIPTS α, β, γ, AND δ RESPECTIVELY INDICATE SIGNIFICANTLY BETTER THAN *RANDOM, TFIDF, WPQ*, AND *LCA*. THE STANDARD DEVIATION OF THE INTERACTION STEPS FOR SUCCESSFUL SESSIONS IS SHOWN BESIDE n_{Step}

		Success rate (r_{success})			Average number of steps (n_{step})			Averaged reward ($\bar{R}$)		
		qry (1)	qry (2)	qry (3)	qry (1)	qry (2)	qry (3)	qry (1)	qry (2)	qry (3)
doc (A)	*random*	50.3%	38.0%	28.8%	2.89 ± 1.44	3.08 ± 1.23	3.11 ± 1.22	0.226	0.140	0.101
	tfidf	57.7%	41.5%	27.7%	2.71 ± 1.33	3.04 ± 1.08	2.94 ± 1.15	0.270^{α}	0.154^{α}	0.104
	wpq	75.4%	43.8%	31.7%	2.61 ± 1.39	3.10 ± 1.26	3.00 ± 1.09	$0.345^{\alpha\beta}$	0.163^{α}	0.115^{α}
	lca	79.1%	39.8%	32.5%	2.60 ± 1.14	2.81 ± 1.33	2.95 ± 1.14	$0.359^{\alpha\beta\gamma}$	0.157^{α}	$0.120^{\alpha\beta}$
	proposed	77.6%	62.6%	43.4%	2.33 ± 1.42	2.78 ± 1.21	2.75 ± 1.31	$0.414^{\alpha\beta\gamma\delta}$	$0.253^{\alpha\beta\gamma\delta}$	$0.173^{\alpha\beta\gamma\delta}$
doc (B)	*random*	46.1%	36.2%	28.2%	2.97 ± 0.91	3.19 ± 1.08	3.17 ± 0.96	0.167	0.123	0.097
	tfidf	52.1%	40.9%	29.5%	2.84 ± 0.98	3.08 ± 1.14	3.23 ± 0.99	0.200^{α}	0.148^{α}	0.103
	wpq	52.4%	44.2%	31.5%	2.91 ± 1.22	3.00 ± 0.88	3.24 ± 0.92	0.206^{α}	$0.160^{\alpha\beta}$	0.106^{α}
	lca	53.5%	42.9%	31.0%	2.98 ± 1.34	3.03 ± 1.10	3.11 ± 0.87	0.205^{α}	0.158^{α}	0.110^{α}
	proposed	78.9%	61.3%	41.8%	2.58 ± 1.06	2.70 ± 0.96	2.69 ± 0.94	$0.336^{\alpha\beta\gamma\delta}$	$0.249^{\alpha\beta\gamma\delta}$	$0.170^{\alpha\beta\gamma\delta}$
doc (C)	*random*	44.8%	35.6%	25.5%	2.99 ± 0.91	3.22 ± 0.82	3.39 ± 0.93	0.163	0.121	0.085
	tfidf	49.6%	40.5%	25.4%	2.78 ± 1.00	3.08 ± 1.08	3.00 ± 1.26	0.195^{α}	0.148^{α}	0.098^{α}
	wpq	52.8%	40.7%	29.6%	2.79 ± 0.81	2.98 ± 1.10	3.08 ± 1.01	$0.204^{\alpha\delta}$	0.147^{α}	0.106^{α}
	lca	46.6%	42.3%	28.4%	2.78 ± 0.83	3.00 ± 1.10	3.26 ± 1.00	0.181^{α}	$0.155^{\alpha\gamma}$	0.097^{α}
	proposed	76.1%	59.7%	40.0%	2.76 ± 1.15	2.66 ± 1.14	2.79 ± 0.90	$0.309^{\alpha\beta\gamma\delta}$	$0.245^{\alpha\beta\gamma\delta}$	$0.157^{\alpha\beta\gamma\delta}$
doc (D)	*random*	41.7%	34.4%	25.2%	3.10 ± 1.09	3.11 ± 0.97	3.44 ± 0.98	0.148	0.118	0.078
	tfidf	47.7%	35.4%	25.2%	2.85 ± 0.86	3.04 ± 1.03	3.31 ± 1.20	0.185^{α}	0.131^{α}	0.086
	wpq	49.5%	39.3%	28.0%	2.91 ± 0.91	3.11 ± 0.88	3.29 ± 0.96	0.186^{α}	0.140^{α}	0.093^{α}
	lca	46.5%	40.3%	28.1%	2.85 ± 0.98	3.08 ± 0.93	3.22 ± 0.88	0.179^{α}	0.142^{α}	0.094^{α}
	proposed	71.0%	57.6%	29.4%	2.60 ± 0.93	3.03 ± 0.92	3.05 ± 0.94	$0.298^{\alpha\beta\gamma\delta}$	$0.218^{\alpha\beta\gamma\delta}$	$0.114^{\alpha\beta\gamma\delta}$

which there are no recognition errors. The assumption with *wpq* that the first m retrieved documents are relevant does not hold when recognition errors yield poor retrieval results with many false alarms. With *lca*, the assumption is that terms which frequently co-occur with the query in the same documents are suitable terms for query expansion; when there are many errors in the recognition results, however, this co-occurrence relationship is no longer reliable.

Fig. 6 details two types of statistics in comparison with the *wpq* and *lca* approaches, for the 163 retrieval sessions: the number of failure retrieval sessions and those for successful retrieval sessions, grouped by the number of interaction steps. These numbers correspond to the doc (C) plus qry (1) case in Table V. With the *wpq* and *lca* methods, 76 and 86 out of the 163 sessions failed, but with the proposed method, only 39 did. Also, many more retrieval sessions could be successfully completed in two steps for the proposed approach (70) as compared with *wpq* (35) and *lca* (32). It is thus clear that the proposed method decreased the number of steps in successful sessions, although the difference in the average number of steps was very small (2.76 versus 2.79 and 2.78, see Table V). Furthermore, note that while with the proposed approach more sessions took five steps or more (3 for 5 steps, 3 for 6 steps, and 3 for 7 steps), these were primarily the sessions which failed with *wpq* and *lca*. This explains the limited improvement in the average number of steps, and also explains significantly improved averaged reward for the proposed approach (0.309 versus 0.204 and 0.181). With the proposed approach, we were not able to guarantee a decrease in n_{step}, but we did ensure an improvement in the averaged reward, which takes into account both r_{success} and n_{step}.

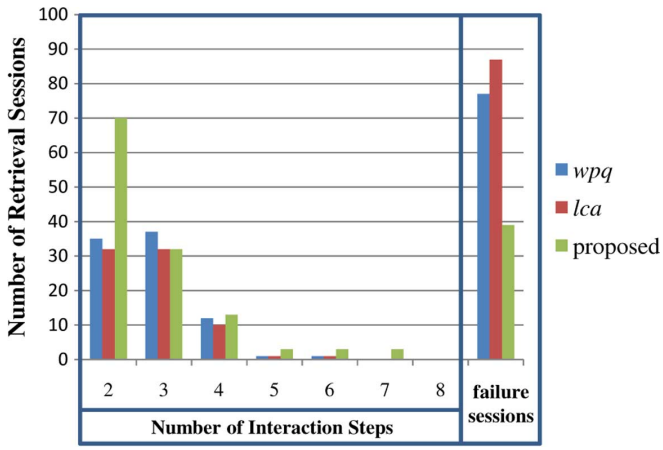

Fig. 6. Numbers of successful sessions (grouped by number of interaction steps) and failure sessions for doc (C) plus qry (1) in Table V.

G. Discussion

In this section, we discuss a few key issues. First, why did the various different algorithms perform so differently in the tasks evaluated here? In comparing the *tfidf* approach with the proposed approach, we note that *tfidf* simply ranks the given key terms using a set of pre-evaluated *tfidf* statistics without taking into account the initial query and the current results. For example, given the query "US President," the term "Software" may still be ranked very high simply because "Software" has a high *tfidf* score, even if it is not very likely to be selected by a user given the query "US President." For the *wpq* and *lca* methods, given the current query "US President," the term "Software" is naturally ranked lower as compared with other more relevant terms such as "Middle East," "Iraq," "China," and

TABLE VI
AVERAGED REWARD WHEN TRAINING AND TEST ARCHIVES ARE MISMATCHED

Averaged Reward ($\bar{R}$)		Training Phase	
		doc (A)	doc (B)
Testing Phase	doc (A)	0.414	0.370
	doc (B)	0.307	0.336

so on. For the proposed approach, the term "Software" is also ranked lower since the system may learn from the training data that given the query "US President," "Software" is almost useless for query expansion to find more specific retrieval results. As another example, given the query "US President," both *wpq* and *lca* naturally rank the key term "American" higher because many documents retrieved with the query "US President" also include "American." For the proposed approach, on the other hand, the system learns from the training data, or simulated users, that "American" almost does not add any extra information given the query "US President," and is therefore not helpful in clarification of the user's need, so the term is naturally ranked much lower. Actually, in the proposed approach, the system looks ahead and plans for the complete retrieval session when ranking the key terms because the key term ranking strategy is trained by many simulated users during reinforcement learning, which takes into account the reward obtainable at the end of the retrieval session. Note that although it is possible to properly rank key terms by simply considering word semantic relationships, here in this work we propose key term ranking methods simultaneously optimizing the successful rate and the number of interaction steps, balancing between two mutually contradictory properties (higher term coverage and higher discriminating power as mentioned in Section II-B). Such purpose is accomplished by looking ahead and planning for the complete retrieval session with reinforcement learning.

Second, why does the proposed approach seem to be relatively robust against recognition errors? One reason is because it learns from training data, or simulated users, and thus learns to some extent the recognition errors that may occur. For example, if word X is frequently recognized as word Y, this error pattern may also be learned, and the retrieval process may therefore not be hurt by such a recognition error. In Table VI, we list the average reward $\bar{R}$ when the training and testing phases use different sets of the document archives with different accuracies. We observe an averaged reward of 0.336 when the key term ranking was both trained and tested with doc (B): this corresponds to that listed in Table V for qry (1). However, when the key term ranking was trained with doc (A), which is 100% accurate, but tested on doc (B), which has recognition errors, the averaged reward degraded to 0.307. This shows that it is worse to train with a perfect archive than to train using an archive with errors, because the errors can be learned and reflected in the key term rankings. On the other hand, when the testing archive was doc (A) (no errors), training with doc (B) (has errors) clearly caused problems and degraded the averaged reward from 0.414 to 0.370. These results show that the recognition errors were learned and reflected in the key term rankings.

Third, we should note that simulated users do not necessarily reflect the information needs and retrieval behavior of real users. The goal here is simply to rank the key terms to help users more quicker find what they are looking for. Even if simulated users are quite different from real users, these experiments show that the key term rankings thus obtained performed reasonably well. In fact, we are confident that given enough log data for real users for use in training, the proposed approach may yield even greater improvements.

VI. CONCLUSION

We proposed an interactive spoken document retrieval approach, in which the retrieval process is organized around a key term hierarchy with MDP-modeled key term rankings. The user selects key terms from this hierarchy to expand his query and find the desired documents more efficiently. Reinforcement learning is performed with simulated users to minimize the interaction steps needed and maximize the retrieval success rate. Significant improvements over existing approaches were observed in preliminary experiments with retrieval sessions provided by real users, even though the MDP was trained with simulated users. A prototype system was also implemented to verify the concept and present the interactive retrieval scenario.

REFERENCES

[1] L. S. Lee and B. L. Chen, "Spoken document understanding and organization," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 42–60, Sep. 2005.
[2] C. Chelba, T. Hazen, and M. Saraclar, "Retrieval and browsing of spoken content," *Signal Process. Mag.*, vol. 25, no. 3, pp. 39–49, May 2008.
[3] S.-Y. Kong, M.-R. Wu, C.-K. Lin, Y.-S. Fu, and L.-S. Lee, "Learning on demand—Course lecture distillation by information extraction," in *Proc. ICASSP*, 2009, pp. 4709–4712.
[4] J. Glass, T. Hazen, S. Cyphers, I. Malioutov, D. Huynh, and R. Barzilay, "Recent progress in the MIT spoken lecture processing project," in *Proc. Interspeech*, 2007, pp. 2553–2556.
[5] J. Kilgour, J. Carletta, and S. Renals, "The ambient spotlight: Queryless desktop search from meeting speech," in *Proc. SSCS*, 2010, pp. 49–52.
[6] S. Whittaker, J. Hirschberg, B. Amento, L. Stark, M. Bacchiani, P. Isenhour, L. Stead, G. Zamchick, and A. Rosenberg, "Scanmail: A voicemail interface that makes speech browsable, readable and searchable," in *Proc. CHI*, 2002, pp. 275–282.
[7] M. Goto, J. Ogata, and K. Eto, "Podcastle: A web 2.0 approach to speech recognition research," in *Proc. Interspeech*, 2007, pp. 2397–2400.
[8] C. Alberti, M. Bacchiani, A. Bezman, C. Chelba, A. Drofa, H. Liao, P. Moreno, T. Power, A. Sahuguet, M. Shugrina, and O. Siohan, "An audio indexing system for election video material," in *Proc. ICASSP*, 2009, pp. 4873–4876.
[9] J.-M. Van Thong, P. Moreno, B. Logan, B. Fidler, K. Maffey, and M. Moores, "Speechbot: An experimental speech-based search engine for multimedia content on the web," *IEEE Trans. Multimedia*, vol. 4, no. 1, pp. 88–96, Mar. 2002.
[10] J. Hansen, R. Huang, B. Zhou, M. Seadle, J. Deller, A. Gurijala, M. Kurimo, and P. Angkititrakul, "Speechfind: Advances in spoken document retrieval for a national gallery of the spoken word," *IEEE Trans. Speech Audio Process.*, vol. 5, no. 1, pp. 712–730, Jan. 2005.
[11] S. W. Lee, K. Tanaka, and Y. Itoh, "Combining multiple subword representations for open-vocabulary spoken document retrieval," in *Proc. ICASSP*, 2005, pp. 505–508.
[12] S. Meng, P. Yu, J. Liu, and F. Seide, "Fusing multiple systems into a compact lattice index for Chinese spoken term detection," in *Proc. ICASSP*, 2008, pp. 4345–4548.
[13] M. Saraclar and R. Sproat, "Lattice-based search for spoken utterance," in *Proc. HLT*, 2004.
[14] H. L. Chang, Y. C. Pan, and L. S. Lee, "Latent semantic retrieval of spoken documents over position specific posterior lattices," in *Proc. SLT*, 2008, pp. 285–288.
[15] Y. C. Hsieh, Y. T. Huang, C. C. Wang, and L. S. Lee, "Improved spoken document retrieval with dynamic key term lexicon and probabilistic latent semantic analysis (PLSA)," in *Proc. ICASSP*, 2006, pp. 961–964.
[16] L. Molgaard, K. Jorgensen, and L. Hansen, "Castsearch—Context based spoken document retrieval," in *Proc. ICASSP*, 2007, pp. 93–96.

[17] I. Ruthven, *Interactive Information Retrieval*. New York: Wiley, 2008, vol. 42, no. 1.
[18] J. Lu, Z. Xie, R. Li, Y. Zhang, and J. Wang, "A framework of CBIR system based on relevance feedback," in *Proc. 2007 Workshop Intell. Inf. Technol. Applicat.*, 2009, pp. 175–178.
[19] J. Cui, F. Wen, and X. Tang, "IntentSearch: Interactive on-line image search re-ranking," in *Proc. MM'08: 16th ACM Int. Conf. Multimedia*, 2008, pp. 997–998.
[20] C. G. M. Snoek, M. Worring, D. C. Koelma, and A. W. M. Smeulders, "A learned lexicon-driven paradigm for interactive video retrieval," *IEEE Trans. Multimedia*, vol. 9, no. 2, pp. 280–292, Feb. 2007.
[21] A. Diriye, S. Zagorac, S. Little, and S. Rüger, "NewsRoom: An information-seeking support system for news videos," in *Proc. Int. Conf. Multimedia Inf. Retrieval MIR'10*, 2010, pp. 377–380.
[22] G. Geisler, G. Marchionini, B. M. Wildemuth, A. Hughes, M. Yang, T. Wilkens, and R. Spinks, "Video browsing interfaces for the open video project," in *Proc. CHI'02: CHI'02 Extended Abstracts on Human Factors in Comput. Syst.*, 2002, pp. 514–515.
[23] T. Misu and T. Kawahara, "Bayes risk-based dialogue management for document retrieval system with speech interface," , vol. 52, no. 1, pp. 61–71, 2010.
[24] T. Kawahara, "New perspectives on spoken language understanding: Does machine need to fully understand speech?," in *Proc. ASRU*, 2009, pp. 46–50.
[25] N. Gupta, G. Tur, D. Hakkani-Tür, S. Bangalore, G. Riccardi, and M. Gilbert, "The AT&T spoken language understanding system," *IEEE Trans. Speech Audio Process.*, vol. 14, no. 1, pp. 213–222, Jan. 2006.
[26] S. Young, "Talking to machines (statistically speaking)," in *Proc. ICSLP*, 2002, pp. 9–16.
[27] Y.-Y. Wang, L. Deng, and A. Acero, "Spoken language understanding," *IEEE Signal Process. Mag.*, vol. 22, no. 5, pp. 16–31, Sep. 2005.
[28] J. Williams and S. Young, "Partially observable Markov decision processes for spoken dialogue systems," *Comput. Speech Lang.*, vol. 21, no. 2, pp. 393–242, 2007.
[29] E. Levin, R. Pieraccini, and W. Eckert, "A stochastic model of human-machine interaction for learning dialogue strategies," *IEEE Trans. Speech Audio Process.*, vol. 8, no. 1, pp. 11–23, Jan. 2000.
[30] 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 *Proc. ASRU*, 2005, pp. 375–380.
[31] Y. C. Pan, J. Y. Chen, Y. S. Lee, Y. S. Fu, and L. S. Lee, "Efficient interactive retrieval of spoken documents with key terms ranked by reinforcement learning," in *Proc. Interspeech*, 2006, pp. 1577–1580.
[32] Y. C. Pan and L. S. Lee, "Simulation analysis for interactive retrieval of spoken documents," in *Proc. SLT*, 2006.
[33] Y. C. Pan, H. L. Chang, and L. S. Lee, "Type-II dialogue systems for information access from unstructured knowledge sources," in *Proc. ASRU*, 2007, pp. 544–549.
[34] B. Billerbeck, "Efficient query expansion," Ph.D. dissertation, RMIT Univ., Melbourne, Australia, 2005.
[35] C.-K. Huang, L.-F. Chien, and Y.-J. Oyang, "Relevant term suggestion in interactive web search based on contextual information in query session logs," *J. Amer. Soc. Inf. Sci. Technol.*, pp. 638–649, 2003.
[36] M. Magennis and C. van Rijsbergen, "The potential and actual effectiveness of interactive query expansion," in *Proc. SIGIR*, 1997, pp. 324–332.
[37] I. Ruthven, "Re-examining the potential effectiveness of interactive query expansion," in *Proc. SIGIR*, 2003, pp. 213–220.
[38] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press, 1998.
[39] S.-C. Chen and L.-S. Lee, "Automatic title generation for Chinese spoken documents using an adaptive k nearest-neighbor approach," in *Proc. Eurospeech*, 2003, pp. 2813–2816.
[40] L. S. Lee, Y. Ho, J. F. Chen, and S. C. Chen, "Why is the special structure of the language important for Chinese spoken language processing?—Examples on spoken document retrieval, segmentation and summarization," in *Proc. Eurospeech*, 2003, pp. 49–52.
[41] S. L. Chuang and L. F. Chien, "Taxonomy generation for text segments: A practical web-based approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 4, pp. 363–396, 2005.
[42] Y.-N. Chen, Y. Huang, S.-Y. Kong, and L.-S. Lee, "Automatic key term extraction from spoken course lectures using branching entropy and prosodic/semantic features," in *Proc. SLT*, 2010.
[43] F. Liu, D. Pennell, F. Liu, and Y. Liu, "Unsupervised approaches for automatic keyword extraction using meeting transcripts," in *Proc. NAACL*, 2009, pp. 620–628.
[44] T. Hofmann, "Probabilistic latent semantic indexing," in *Proc. SIGIR*, 1999, pp. 50–57.
[45] R. V. Hogg and A. T. Craig, *Introduction to Mathematical Statistics*, 4th ed. New York: Macmillan, 1978.
[46] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, "Query expansion by mining user logs," *IEEE Trans. Knowl. Data Eng.*, vol. 15, no. 4, pp. 829–839, Jul.-Aug. 2003.
[47] S. E. Robertson, "On term selection for query expansion," *J. Documentat.*, vol. 46, pp. 129–146, 1990.
[48] J. Xu and W. B. Croft, "Improving the effectiveness of information retrieval with local context analysis," *ACM Trans. Inf. Syst. (TOIS)*, vol. 18, no. 1, pp. 79–112, 2000.
[49] S. Yu, D. Cai, J.-R. Wen, and W.-Y. Ma, "Improving pseudo-relevance feedback in web information retrieval using web page segmentation," in *Proc. 12th Int. Conf. World Wide Web (WWW'03)*, Budapest, Hungary, 2003, pp. 11–18.

Yi-Cheng Pan was born in 1978. He received the B.S., M.S., and Ph.D. degrees in computer science and information engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2000, 2002, and 2008, respectively.

From January 2004 to January 2008, he was a Research Assistant supervised by Prof. L.-F. Chien with the Institute of Information Science, Academia Sinica, Taipei. From October 2008 to July 2009, he was a Postdoctoral Researcher in Prof. S. Furuis' Laboratory, Tokyo Institute of Technology, Tokyo, Japan. He is currently an Engineer with MediaTek, Inc., Hsinchu, Taiwan. His research has been focused on speech recognition, information retrieval, and spoken dialogue systems.

Hung-Yi Lee was born in 1986. He received the B.S. and M.S. degrees in electronic engineering and communication engineering from National Taiwan University (NTU), Taipei, Taiwan, in 2008 and 2010, respectively. He is currently pursuing the Ph.D. degree in the Department of Communication Engineering, NTU.

His research focuses on speech information retrieval.

Lin-Shan Lee (F'03) received the Ph.D. degree in electrical engineering from Stanford University, Stanford, CA.

He has been a Professor of electrical engineering and computer science at National Taiwan University (NTU), Taipei, Taiwan, since 1982 and holds a joint appointment as a Research Fellow of Academia Sinica, Taipei. His research interests include digital communications and spoken language processing. He developed several of the earliest versions of Chinese spoken language processing systems in the world including text-to-speech systems, natural language analyzers, dictation systems, and voice information retrieval systems.

Dr. Lee was Vice President for International Affairs (1996–1997) and the Awards Committee chair (1998–1999) of the IEEE Communications Society. He was a member of the Board of International Speech Communication Association (ISCA 2002–2009), a Distinguished Lecture (2007–2008) and a member of the Overview Paper Editorial Board (since 2009) of the IEEE Signal Processing Society, and the general chair of ICASSP 2009 in Taipei. He has been a fellow of ISCA since 2010, and received the Meritorious Service Award from IEEE Signal Processing Society in 2011.