Interactive Spoken Content Retrieval with Different Types of Actions Optimized by a Markov Decision Process

Tsung-Hsien Wen[1], *Hung-Yi Lee*[2], *and Lin-Shan Lee*[1,2]

[1]Graduate Institute of Electrical Engineering,
[2]Graduate Institute of Communication Engineering,
National Taiwan University, Taipei, Taiwan
r00921033@ntu.edu.tw, lslee@gate.sinica.edu.tw

Abstract

Interaction with user is specially important for spoken content retrieval, not only because of the recognition uncertainty, but because the retrieved spoken content items are difficult to be shown on the screen and difficult to be scanned and selected by the user. The user cannot playback and go through all the retrieved items and then find out they are not what he is looking for. In this paper, we propose a new approach for interactive spoken content retrieval, in which the system can estimate the quality of the retrieved results, and take different types of actions to clarify the user's intention based on an intrinsic policy. The policy is optimized by a Markov Decision Process (MDP) trained with Reinforcement Learning based on a set of pre-defined rewards considering the extra burden given to the user.

Index Terms: Interactive SDR, MDP, Reinforcement Learning

1. Introduction

Numerous successful spoken dialogue systems have been developed in the past decades. They were usually designed for a specific application domain, such as airline itinerary planning [1], train information management [2], and tutoring systems [3]. Many of such systems are based on a well-defined database at the back-end and a statistical dialogue manager such as those similar to a Markov Decision Process (MDP). On the other hand, interactive information retrieval (IIR) [4, 5] offers a scenario for human-machine interaction to help the machine clarify user's intention during retrieval. Through the interactions, the user may transmit better information to the machine regarding to what he is looking for, and the machine may therefore more accurately return what the user wants. However, for IIR research, little efforts have been made in borrowing the experiences and expertise from spoken dialogue systems so far.

A good example for IIR system is "Dialogue Navigator for Kyoto City" [6, 7] , which used a Bayes risk-based dialogue manager to offer an efficient interaction interface to find information about Kyoto city. Interactive retrieval is specially important for spoken content, not only because recognition errors produce high degree of uncertainty for the content, but because the spoken content is difficult to be shown on the screen and difficult to be scanned and selected by the user. The user cannot simply playback and go through all the retrieved items and then choose the one he is looking for. In a recent work, the Markov decision process (MDP) popularly used in spoken dialogue systems was used to help the user select keyterms in the IIR process of a broadcast news browser [8]. In this approach, in each iteration the system returned not only a list of retrieved items, but a set of keyterms ranked by MDP for the user to select if the user is not satisfied with the retrieved items. But when the retrieved result is poor, the user still needs to take long time to go through the retrieved items to find that the result is unsatisfactory and then selects the keyterm for the next iteration. A much better scenario would be as follows. When the retrieved result is poor, the machine can be smart enough to actively interact with the user to clarify the user's intention.

In this paper, we propose a new approach for interactive spoken content retrieval, in which the system can estimate the quality of the retrieved results realizing the above scenario, and take different actions to interact with the user to improve the retrieved results. The actions are optimized by MDP trained with reinforcement learning optimizing a set of pre-defined rewards considering the extra burden given to the user.

2. Scenario of the proposed approach

U1:	US President, please.
S1:	Your query is ambiguous. Anything more?
U2:	Diplomatic issue.
S2:	Persian Gulf?
U3:	No.
S3:	Please view this list and select one item relevant to your need.
U4:	(Pick the doc at rank 6th, "Obama: We Welcome China's Rise, January 19, 2011 3:47 PM, CBSNEWS")
S4:	(Show the list) Here is the result.

Figure 1: A scenario example between system (**S**) and user (**U**).

Fig. 1 is a possible interaction scenario for a broadcast news retrieval system. Suppose now the user is looking for the news about the meeting of US President Barack Obama with the leader of China, Hu Jintao. But the user does not know what is in the back-end archive and thus is not able to formulate an efficient query for the propose. Instead, he enters the short query "US President" (U1). The query "US President" is ambiguous since there are too many documents in the archive related to "US President". As a result, the retrieved list may be very noisy. However, if the system is able to interact with the user as shown in Fig. 1, the system finds this situation and thus asks the user for further information (S1), and receives the next instruction of "Diplomatic issue" from the user (U2). Since many news items related to "US President" and "Diplomatic issue" are about Persian Gulf, so the system further asks if the user wants news related to "Persian Gulf" and gets the answer "No" (S2, U3). This answer narrows down the target, and therefore the system offers a list of possible item examples for the user to select (S3). With the selection (U4) the system then has enough information to retrieve the documents the user wants. Therefore, the retrieval results are presented to the user (S4).

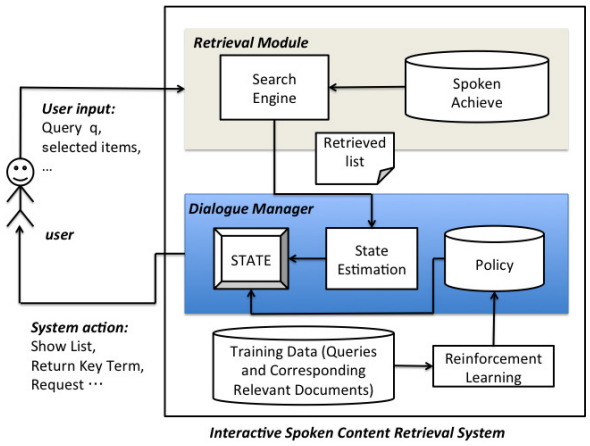

Figure 2: Block diagram of the proposed system.

3. Proposed Approach

3.1. Retrieval Module

The upper part of Fig. 2 is the system's retrieval module, or the conventional spoken content retrieval. If a query q is entered, the module assigns a relevance score $r_q(d)$ to each spoken document d in the achieve, which represents the degree of relevance between the query q and spoken document d. In most events, the documents d are first transcribed into lattices by a recognition engine and then indexed in some way, for example, in form of position-specific posterior lattices (PSPLs). The matching scores accumulated for all n-grams within the query q evaluated from the indices are weighted and summed to be the relevance score of document d with respect to query q, $r_q(d)$.

3.2. Action Set

There are four possible system actions defined here: *(a) Return Documents*, *(b) Return Keyterm*, *(c) Request*, and *(d) Show List*. After a query q is submitted, the retrieval module searches through the indices using the query q, and each document d in the corpus is assigned a relevance score $r_q(d)$, which is taken as the score for each document d at retrieval turn 0, $R_0(d)$ ($R_0(d) = r_q(d)$). At each turn i, the system either (1) shows the retrieved results ranked according to $R_{i-1}(d)$ and ends the retrieval session (action *(d) Show List*), or (2) interacts further with the user and updates the relevance score of each document as $R_i(d)$ if one of the other three actions, *(a) Return Documents*, *(b) Return Keyterm*, or *(c) Request*, are selected.

(a) Return Documents: the system returns the current retrieved list ranked decreasingly by $R_{i-1}(d)$, and asks the user to view the document list from the top and select a relevant document as the feedback. The system then extracts a set of terms $\mathcal{T}$ which is the M most important terms in the user-selected document based on TF-IDF weights, and each term t in $\mathcal{T}$ is used as a query submitted to the retrieval engine. The relevance score $R_{i-1}(d)$ is then updated to $R_i(d)$ as

$$R_i(d) = \alpha R_{i-1}(d) + \frac{\beta}{M} \sum_{t \in \mathcal{T}} r_t(d), \tag{1}$$

where $r_t(d)$ is the relevance score obtained with term t taken as the query, and α and β are weighting parameters.

(b) Return Keyterm: the system asks the user if a keyterm t^* is relevant,

$$t^* = \arg\max_t \sum_{d \in \mathcal{D}} f(d,t) ln(1 + idf(t)), \tag{2}$$

where $\mathcal{D}$ is the top N documents with the highest $R_{i-1}(d)$, $f(d,t)$ is the term frequency of term t in document d, and $idf(t)$ is the inverse document frequency of term t. If the user judges that term t^* is relevant, then

$$R_i(d) = \alpha R_{i-1}(d) + \beta r_{t^*}(d). \tag{3}$$

Otherwise, the system takes the term t^* in a negative way,

$$R_i(d) = \alpha R_{i-1}(d) - \gamma r_{t^*}(d), \tag{4}$$

where γ is another weighting parameter.

(c) Request: the user is asked to provide another query $\hat{t}$, and the relevance score $R_i(d)$ for each document d is

$$R_i(d) = \alpha R_{i-1}(d) + \beta r_{\hat{t}}(d). \tag{5}$$

Each action above has different properties and should be chosen carefully based on the particular situation. For example, *(a) Return Documents* can modify the query in a significant way, while *(b) Return Keyterm* performs minor adjustment on the retrieved results with a very low user cost since the user only has to answer yes or no to the system. *(c) Request* may offer the system the most information but costs high for the user. Therefore, how to efficiently combine these different choices is the main issue in our system.

3.3. State Space

The action the system should take at each turn is decided by its intrinsic policy π, which is a function whose input is the dialogue state and output is the action to be taken. We use two variables to define the state of the system at turn i. The first is the quantized retrieval performance of the result ranked by $R_{i-1}(d)$. Here we quantize the values of a selected evaluation metric (for example, mean average precision (MAP)) into several bins to indicate different levels of quality of retrieved results, or different states. This is reasonable because the quality of the retrieved results should be closely related to the confidence of the system about estimating the user's intention. The other variable used to define the state is the number of turns that have taken place in the dialogue so far, that is, the value of i. In reality, the retrieval performance can be judged only by the user, that is, the system can never exactly know how closely the current retrieval result matches what the user is looking for, so the system never knows the true state it is in, but can estimate it. This is why a state estimation module is in the middle of our dialogue manager of Fig. 2, which is described in Sec. 3.6.

3.4. Reward

When an action is taken, the system obtains a reward, and the Final Return the system earns for a retrieval session, $G^{\pi}_{s_0}$, is the summation of the rewards for all the actions taken following the policy π in the session,

$$G^{\pi}_{s_0} = \sum_{a \in A^{\pi}_{s_0}} C(a), \tag{6}$$

where s_0 is the initial state, $A^{\pi}_{s_0}$ is the sequence of actions to be taken at the states following s_0 along the session based on the policy π, and $C(a)$ the reward for action a. In the proposed approach here, negative rewards, or costs, are assigned to actions *(a) Return Documents*, *(b) Return Key-term*, and *(c) Request* since they all involve effort from the user side. On the other hand, the system obtains positive reward when the action *(d) Show List* is taken, which is the gain of the retrieval evaluation metric via interaction. The reward of *Show List* is defined as $\lambda(E_f - E_0)$, where E_0 is the value of a certain evaluation

metric (e.g. mean average precision(MAP) or similar) of the first-pass retrieved results ranked by $R_0(d)$, E_f a certain evaluation metric for the retrieved results of the final state in which *Show List* action is taken to finish the retrieval session, and λ the trade-off parameter between user efforts and retrieval quality, where a smaller λ indicates the system prefers to minimize the user efforts than maximize the retrieval result quality. This is because rewards for all other actions are costs for user effort. In the experiments below, we tested two sets of λ settings that justify the generalization of this work.

3.5. Policy Training

As long as a set of training data including queries and the corresponding relevant documents is available, reinforcement learning can be used to train a policy from the training data. Here we adopt the ϵ-soft on-policy Monte-Carlo control algorithm [9]. This algorithm updates the policy iteratively from an initial policy π^0. At iteration n, a training query q is first randomly selected from the training set as the input query for the retrieval session Z_n. The system generates Z_n as the following. For each traversed state s^*, the system takes action a^* following π^{n-1} ($a^* = \pi^{n-1}(s^*)$), the policy obtained in the last iteration, with probability $1-\epsilon$, but randomly takes a different action from the action set other than a^* with probability ϵ, which is usually set close to zero [1]. After the session Z_n reaches its final state, the Q-function for value estimation in reinforcement learning for each state-action pair (s, a) occurring in Z_n is updated as

$$Q^n(s,a) = \frac{(n-1)Q^{n-1}(s,a) + G_s^{\pi^{n-1}}}{n}, \qquad (7)$$

where $G_s^{\pi^{n-1}}$ is the Final Return the system obtains in session Z_n started with state s as defined in (6). Note that the state of the system is known because the relevant documents for query q is known in the training set. The policy π^n is then

$$\pi^n(s) = \arg\max_a Q^n(s,a). \qquad (8)$$

If π^n is equal to π^{n-1}, then the policy training converges and we use policy π^n for testing [9]. Such a training process is possible because in the training set the relevant documents for a query is known, and thus the state is known to the system. During testing, however, the state is unknown because the system does not know which documents are truly relevant to the user's query. Below we thus describe some approaches to estimate the system state.

3.6. State Estimation

In order to know the state of the system at turn i, the system should estimate the retrieval performance level for the results ranked by $R_{i-1}(d)$. In this work, we combine different pre-retrieval and post-retrieval predictors for this purpose. These predictors include clarity score, ambiguity score [10], similarity of the query and the collection [11], query length, the top-N similarity score between query and retrieved list as well as its statistical indicators such as mean and variance. We consider each performance level as a class and use these predictors as features to train an SVM multi-class classifier from the training set. However, this is a very difficult classification problem since we need to "guess" the user intent only from the predictor, and direct SVM training shows very bad results because of the problem of insufficient information of the retrieved list. As a consequence, we adopt a heuristic rule to first cluster training data into several local clusters based on two criteria: one is the relevance score of the top-1 ranking document and the other is the dialogue turn, and then train an SVM multi-class classifier for each cluster. In testing phase, we use the two criteria to first assign a cluster for the retrieved list, then estimates its performance level by the corresponding SVM multi-class classifier.

[1] If actions that violate the policy are never taken, some state-action pairs will never be explored during the training process.

4. Experiments

4.1. Experimental Setup

In the experiments, we used broadcast news stories in Mandarin Chinese as the spoken document archive to be retrieved from. There were a total of 5047 news stories, recorded in Taipei from 2001 to 2003 with a total length of 96 hours. For recognition we used a 60K-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 in order to conduct evaluations for different recognition accuracies. As listed below, we used two different recognition conditions for the spoken documents: *Doc (I)*: The spoken documents were recognized by Acoustic models with 8 Gaussian mixtures per state trained on a corpus of 24.5 hours of broadcast news different from the archive tested here, with character accuracy 45.64%. *Doc (II)*: Same as *Doc (I)*, but with 64 Gaussian mixtures per state, with character accuracy 52.15%.

Although the evaluation metric can vary from task to task, Mean Average Precision (MAP) was selected for evaluation here, since it is widely adopted as an overall statistical performance measure for IR. As mentioned in section 3.3, the performance level and number of turns are two variables defining the states. We quantized the MAP scores of the retrieval results into 4 bins as 4 performance level. The variable representing the number of turns was ranged from 1 to 5, and when the number of turns exceeded 5, we set the variable as 5. The policy training was started with a plain policy which took the action *Show List* at any state. With the consideration of the burden of each action given to the user, the costs of actions were set empirically.

163 sets of query and relevant document set were provided by 22 graduate students, each including a query in text form and its corresponding relevant documents. The number of relevant documents for each query ranged from 1 to 50 with an average of 19.5, and the query length ranged from 1 to 4 Chinese words. In the experiments below, we simulated the interaction between user and machine. When the system took the action *Return Documents*, the simulated user viewed the list from the top and chose the first relevant document. For the action *Return Keyterm*, the simulated user replied "YES" if the keyterm occurred in more than 50% of the relevant documents. For the action *Request*, the simulated user entered a term t^* as the new query, $t^* = \arg\max_t \sum_{d \in \mathcal{D}_\mathcal{I}} f(d,t) ln(1 + idf(t))$ where D_I is the relevant document set. 4-fold cross validation was performed in the following experiments, that is, in each trail 3 out of 4 query folds were used for policy training, and the remaining 1 fold for testing.

4.2. Experimental Results

Experimental results are shown in Table 1. The two Sections *Doc (I)* and *(II)* are the results based on the transcriptions of different acoustic models as mentioned above in Sec. 4.1, and we tested the setting of λ with 1000 and 2000. In Table 1, both the results in terms of MAP and Final Return is defined in (6) are reported. We compare the proposed approach with 4 baselines (rows (1) to (4)). Row (1) is the first-pass results without any interaction, while rows (2), (3) and (4) are respectively the results for the system which took a fixed action *Return Documents*, *Return Keyterm*, or *Request* n times and then *Show list*. The value

Table 1: Mean average precision (MAP) and Final Return(F.Return) for different settings of λ and different recognition accuracies.

		Doc (I)				*Doc (II)*			
		λ=1000		λ=2000		λ=1000		λ=2000	
Policy		MAP	F.Return	MAP	F.Return	MAP	F.Return	MAP	F.Return
Baseline	(1) No Interaction	0.3703	–	0.3703	–	0.4335	–	0.4335	–
	(2) *Return Documents*	0.3341	-56.31	0.3341	-92.61	0.3807	-72.87	0.3807	-125.7
	(3) *Return Key-term*	0.3693	-11.23	0.3693	-12.28	0.4283	-15.49	0.4283	-20.79
	(4) *Request*	0.4241	4.18	0.4241	57.99	0.4890	5.72	0.4890	61.08
MDP	**(5) Oracle**	**0.4607**	**46.75**	**0.4735**	**139.73**	**0.5249**	**40.63**	**0.5267**	**124.71**
	(6) Estimate	**0.4335**	**19.92**	**0.4558**	**91.25**	**0.4920**	**23.30**	**0.4975**	**84.05**

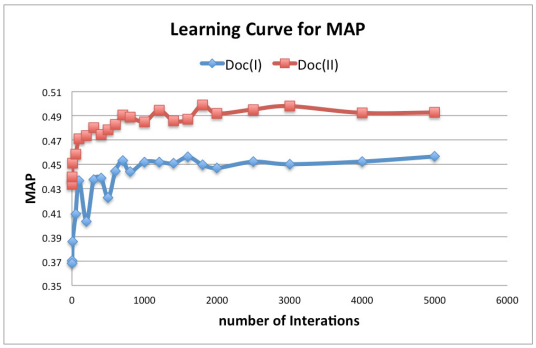

Figure 3:
Mean Average Precision (MAP) for the proposed approach with estimated states under different number of training iterations.

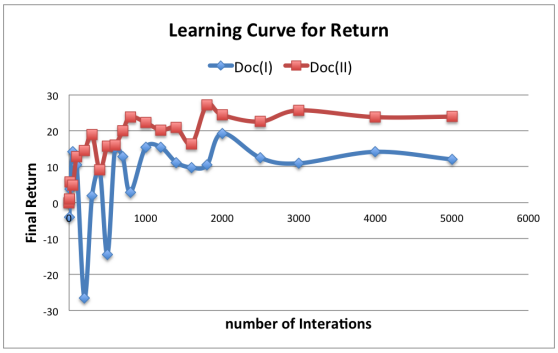

Figure 4:
Final Return for the proposed approach with estimated states under different number of training iterations.

of n was tuned to give the best results. Rows (5) and (6) are the results for the proposed approach after the system performance converges. Row (5) is the results that the state at each turn was completely known to the system, which can be considered as the performance upper bound for the proposed method. Row (6) is the results with estimated states. Among all the baselines listed in Table 1, we found that only *Request* achieved improvements in terms of MAP and Final Return (rows (4) vs (1)), while taking only *Return Documents* or *Return Keyterm* could not offer any improvements compared with first-pass results (rows (2) and (3) vs (1)). However, the oracle results (row (5)) show that under the condition of perfect state information, using MDP to model the interaction mechanism attained a very significant improvements both in terms of MAP and Final Return. For the results with estimated states, due to the unavoidable imperfect state prediction, performance is poorer than oracle (rows (6) vs (5)) but actually much better than any other baseline policies listed in Table 1 (rows (6) vs (1), (2), (3) and (4)). Also note that different recognition accuracies and λ settings exhibit the same trends in out experiments.

Fig. 3 and Fig. 4 respectively show the learning curves for the MAP and Final Return for the proposed method with estimated states under different training iterations. Despite of some jitters in the early phase of training, both the MAP and Final Return grew gradually during learning, and then started to saturate around 2000 iterations.

5. Conclusion

Due to the recognition uncertainty and browsing difficulty for spoken contents, interactive spoken content retrieval comes into play a significant role. In this paper, we propose a new approach to model the interactive spoken content retrieval process as a Markov decision process (MDP) whose policy is optimized by Reinforcement Learning based on the trade-off between the quality of the retrieved list and the extra burden imposed to the user. The quality of the retrieval result is used to estimate the state of the system, and several query performance predictors trained by multi-class SVM are used to predict the system's current state. Our experiments unveil that this system can learn how to act properly given any state from a bunch of simulated training episodes and a completely plain policy.

6. References

[1] Stephanie Seneff and Joseph Polifroni, "Dialogue management in the mercury flight reservation system," in *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems*, 2000.

[2] Lamel Bennacef, L. Lamel, S. Bennacef, J. L. Gauvain, H. Dartiguesy, and J. N. Tememy, "User evaluation of the mask kiosk," in *in Proceedings of ICSLP '98*, 1998.

[3] Diane J. Litman and Scott Silliman, "Itspoke: an intelligent tutoring spoken dialogue system," in *Demonstration Papers at HLT-NAACL 2004*, 2004.

[4] David Robins, "Interactive information retrieval: Context and basic notions," *Informing Science Journal*, 2000.

[5] Ian Ruthven, "Interactive information retrieval," *Annual Review of Information Science and Technology*, 2008.

[6] Teruhisa Misu and Tatsuya Kawahara, "Bayes risk-based dialogue management for document retrieval system with speech interface," *Speech Commun.*, January 2010.

[7] T. Misu and T. Kawahara, "Speech-based interactive information guidance system using question-answering technique," in *Acoustics, Speech and Signal Processing, ICASSP*, 2007.

[8] Y.-C. Pan, H.-Y. Lee, and L.-S. Lee, "Interactive spoken document retrieval with suggested key terms ranked by a markov decision process," 2012, Audio, Speech, and Language Processing.

[9] Richard S. Sutton and Andrew G. Barto, "Reinforcement learning: An introduction," *Cambridge Journal*, 1999.

[10] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft, "Predicting query performance," 2002, SIGIR '02, ACM.

[11] Ying Zhao, Falk Scholer, and Yohannes Tsegay, "Effective pre-retrieval query performance prediction using similarity and variability evidence," in *Advances in Information Retrieval*. 2008.