Combining Slot-based Vector Space Model for Voice Book Search

Cheongjae Lee and Tatsuya Kawahara and Alexander Rudnicky

Abstract We describe a hybrid approach to vector space model that improves accuracy in voice search for books. We compare different vector space approaches and demonstrate that the hybrid search model using a weighted sub-space model smoothed with a general model and a back-off scheme provides the best search performance on natural queries obtained from the Web.

1 Introduction

The book shopping domain poses interesting challenges for spoken dialog systems as the core interaction involves search for an often under-specified item, a book for which the user may have incomplete or incorrect information. Thus, the system needs to first identify a likely set of candidates for the target item, then efficiently reduce this set to match that item or items originally targeted by the user. This part of the process is characterized as "voice search" and several such systems have been described (Section 2). In this paper we focus on the voice search algorithm and specifically on two sources of difficulty: users not having an exact specification for a target, and queries being degraded through automatic speech recognition (ASR) and spoken language understanding (SLU) errors.

Cheongjae Lee
Academic Center for Computing and Media Studies, Kyoto University, Kyoto, Japan, e-mail: lcj80@ar.media.kyoto-u.ac.jp

Tatsuya Kawahara
Academic Center for Computing and Media Studies, Kyoto University, Kyoto, Japan, e-mail: kawahara@ar.media.kyoto-u.ac.jp

Alexander Rudnicky
Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, USA, e-mail: alex@cs.cmu.edu

One of the major problems is that users may not have the exact information about a book. For example, if the correct book title is "ALICE'S ADVENTURES IN WONDER-LAND AND THROUGH THE LOOKING-GLASS, the user may not remember the entire title and might say "I don't know the whole title but it's something like ALICE ADVENTURE". We previously found that 33% of 200 respondents in a survey did not have the complete information. Moreover, many titles are simply too long to say even though the user might know the exact title (in our database the longest title has 38 words). Thus, users often provide a few keywords instead of exact title. There are additional peculiarities. For instance, the title "MISS PARLOA'S NEW COOK BOOK" is a book by Ms. Parloa in the cookbook category, but this title contains its author's name and the category. These problems may cause degradation in a system that attempts to parse the input. The problem is exacerbated by the large number of eBooks as well as inconsistencies in the database [1].

To address these problems, this paper presents a robust search algorithm based on sub-space models for voice search applications. Specifically, we propose a hybrid approach which combines slots-based models with the general models as a back-off.

2 Related Work

Voice search [6] has been used in various applications: automated directory assistance system [8], consumer rating system [9], multimedia search [4], and book search [3]. Early voice search systems primarily focused on issues of ASR and search problems in locating business or residential phone listings [8]. Then, it has been extended to general web search such as Google's. Recent voice search systems have been applied to search for entries in large multimedia databases [4]. Although a simple string matching technique was used to measure the similarity of an ASR output string to entity values in the database [3], vector space models (VSM) have been widely used [8, 9]. We also propose a hybrid search model using slot-based VSMs and a back-off scheme to improve the search accuracy.

3 Data Collection

3.1 Backend Database

Our system contains a relational database (RDB) consisting of 15,088 eBooks, sampled randomly from the Amazon Kindle Book website. For each book we harvested 17 attributes including its title, authors, categories, price, sales rank. Although many attributes can be used to search for appropriate books, it is not necessarily practi-

[1] Currently, 800,126 eBooks are available in the Amazon Kindle Store [`http://www.amazon.com/Kindle-eBooks/`, retrieved January 6th, 2011]

Slots	Title	Author	Category
Max. Length	38	10	5
Avg. Length	6.99	2.25	1.53
Voca. Size	13708	8159	1002

Table 1 Statistics of the book database.

cal to handle all possible attributes. To define the set of slots for use in the system, we surveyed 221 persons on which information they typically have when they buy eBooks. Note that the respondents had previously bought eBooks.The top three were title (31.99%), authors (26.10%), and category (14.73%). Consequently, we focus on these three slots. Table 1 shows the statistics of the book database used in our system. These contribute 20,882 unique words to the system vocabulary.

3.2 Query Collection using Amazon Mechanical Turk

A key challenge in building a voice search system is defining a habitable user language prior to the point at which a prototype system is available to collect actual user data. Often the procedure consists of the developer and a few other volunteers generating likely inputs as language data. This approach necessarily introduces a sampling bias. We sought to improve this sample diversity by using the Amazon Mechanical Turk (MTurk) service to obtain user utterances at a low cost. MTurk is an on-line marketplace for human workers (turkers) who perform "human intelligence tasks" (HITs) in exchange for small sums of money [2].

We created HITs to elicit utterances, providing metadata consisting of title, authors, and a category. Turkers were asked to formulate a response to the question "how can I help you?" posed by a hypothetical human bookstore clerk. A typical query might be "I AM LOOKING FOR ALICE IN WONDER-LAND BY CARROLL". Although we asked that the turkers think in terms of a spoken query, it is important to note that the queries collected were written, not spoken.

In addition, although we have been developing the whole voice search system in which users can find the target book by interacting with the system, the current queries were not in the context of a dialog system, but appeared at the first turn in a dialog. We have focused on the first user's turn because the first queries should be well-processed for efficient interaction.

4 Book Search Algorithm

The search problem in our system is to return a relevant set of books given noisy queries. In this section, we describe how to search for relevant books in voice book search.

4.1 Baseline Vector Space Model (VSM)

Our vector-space search engine uses a term space, where each book is represented as a vector with specific weights in a high-dimensional space (v_i). A query is also represented as the same kind of vector (v_q). The retrieved list of book is created by calculating the cosine similarity, $s(v_q, v_i)$ between two vectors as follows:

$$s(v_q, v_i) = \frac{v_q \cdot v_i}{\|v_q\| \|v_i\|} \tag{1}$$

If the vectors are normalized, it is possible to compute the cosine similarity as the dot product between the unit vectors.

$$s(v_q, v_i) = \hat{v}_q \cdot \hat{v}_i \tag{2}$$

This formulation allows for rapid search, important as there are many vectors to compute for each query.

We use stemming to compact the representation, but we do not eliminate stop words as some stop words are necessary and meaningful for identifying relevant books. For example, some titles consist of only stop words such as "YOU ARE THAT" and "IT". They will not be indexed correctly if stop words are filtered out.

There are several different ways of assigning term weights but not all are appropriate for this task. For example, TFxIDF does not work well for book search since most values and queries are too short to estimate reliable weights. We used a simple term count weight to represent term vectors; weights indicating the occurrence count for a given term.

In the conventional single vector space model (here, SVSM), all terms in different slots are indexed together over a single term space and every term is equally weighted regardless of its slot name. In such a model, slot names may not be necessary for book search because all query terms are integrated into a single query vector. This model can be robust against SLU errors in which the slot names are incorrectly extracted. This model might also be adequate for books in which the title includes its author's name and category. However, it cannot capture inter-slot relationships. For example, when some users who provides a mixed category query ("A MYSTERY BY CHRISTIE").

4.2 Multiple VSM

We also consider a multiple VSM model (MVSM) in which each slot j is independently indexed over sub-spaces, and each slot-based model is then interpolated with slot-specific weights, w_j, as follows:

$$l^* = \operatorname*{argmax}_l \sum_j w_j \cdot s_j(v_{qj}, v_j) \tag{3}$$

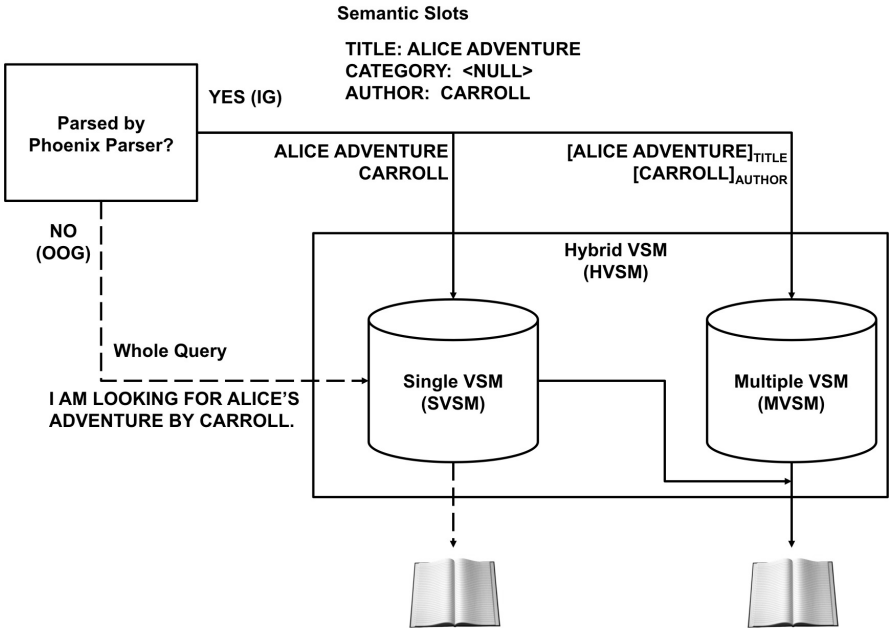

Fig. 1 The strategy of database search (IG: In-grammar, OOG: Out-of-grammar).

Each query is parsed using the Phoenix semantic parser [7]. The slot-based vector v_{qj} for MVSM is generated by slot values extracted by the parser (Figure 1).

Although the interpolation weights w_j can usually be set empirically or by using held-out data, these weights can be modified based on a user's preferences or on confidence scores derived from speech recognition. For instance, if the slots were unreliable, the slot values could be less weighted for the book search. In this work, the weights w_j were set based on the slot preferences that users expressed in our survey (see Section 3.1).

For unfilled slots, the weight is set to 0 and the weights are renormalized dynamically according to the current slot-filling coefficient (f_j) that is assigned a value of one if the slot name j is filled, as follows:

$$\hat{w_j} = \frac{\sum_k f_k \cdot w_k}{f_j \cdot w_j} \tag{4}$$

MVSM can be easily tuned to improve the quality of list generation. Nevertheless, incorrect word-to-slot mapping could degrade search performance relative to SVSM since MVSM relies on this mapping.

4.3 Hybrid VSM and a Back-off Scheme

Finally, we propose a hybrid search model (HVSM) in which SVSM and MVSM are linearly interpolated with a specific weight. We expect that HVSM can compensate for the individual drawbacks of the SVSM and MVSM models, though at the cost of additional computation.

Some queries may be out-of-grammar (OOG) and do not result in a parse. In this case, SVSM can be used as a back-off search model in which all terms in the query are converted into an input vector without the slot information allowing search to be carried out.

5 Search Evaluation

5.1 Experiment Set-up

We use two evaluation metrics widely used in information retrieval. One is precision at n ($P@n$), which represents the number of correct queries having the answer in the top n relevant items divided by the total number of queries. The other is mean reciprocal rank (MRR), which indicates the average of the reciprocal ranks of search results for a sample of queries [5].

In reality there may be multiple correct answers in a list, when users do not have the exact book in their mind. For example, some users can search for any fictions without an exact book in their mind. Because it is difficult to automatically determine the relevance relationship between the queries and the lists, we identified a single correct book corresponding to each query.

We collected 948 textual queries in MTurk (Section 3.2). We then manually created a grammar covering observed query patterns (e.g., "I'D LIKE A BOOK BY [AUTHOR]"), but sub-grammars for the slot values were automatically generated from the book database. To define the sub-grammars, the book titles were tokenized into a bag of words; title queries have many combinations of words regardless of their orders and grammars because users can say content words (e.g. 'ALICE', 'ADVENTURE', 'WONDERLAND', etc) without functional words (e.g. 'IN', 'OF', 'THROUGH', etc). The author names were divided into the first name, the middle name, and the last name because users man say either the full name or a partial name. For example, either 'LEWIS', 'CARROLL', or 'LEWIS CARROLL' may be spoken when users are looking for books by 'LEWIS CARROLL'. Note that the 661 books used for collecting the queries were different from the books used to make the evaluation sub-grammars.

Parsing with the resulting grammar does not always map slot information correctly, even if the query is fully parsed; therefore, SLU will introduce errors even with correct input. Out of 948 test queries, 392 queries had no parse results due to lack of coverage. The F_1 score of the semantic parser on 556 parsed queries is

Type	#Queries	Avg. Words	Avg. Slots
Parsed	556	12.01	1.97
Unparsed	392	15.77	2.10
Total	948	13.56	2.02

Table 2 Statistics of the queries collected in MTurk. #Queries means the number of queries given the query type. Avg. Words and Avg. Slots represent the average number of words and slots in the queries, respectively.

Query Type	SVSM (naïve)		SVSM (parsed)		MVSM		HVSM	
	P@100	*MRR*	*P*@100	*MRR*	*P*@100	*MRR*	*P*@100	*MRR*
Parsed	0.8849	0.6048	0.9137	0.7080	0.8327	0.6905	0.9335	0.7710
Unparsed	0.8087	0.5386	-	-	-	-	-	-
Total	0.8534	0.5774	0.8703	0.6380	0.8228	0.6296	0.8819	0.6763

Table 3 Evaluation results on textual queries (WER=0%). SVSM (naïve) refers to SVSM with the whole query.

75.20% in which the slot values are partially matched by using the cosine similarity between the reference and the hypothesis because the slot values do not necessarily exactly match the information in the database. For instance, “ALICE” or “ADVENTURE” may be individually meaningful to search for the relevant books although “ALICE’S ADVENTURE” might not be parsed from the utterance “I AM LOOKING FOR ALICE’S ADVENTURE BY CARROLL”. Table 2 shows the statistics of our test queries.

5.2 Evaluation on Textual Queries

To evaluate the search performance on the textual queries collected through MTurk, we used parsed, unparsed, and total queries (Table 3). First, these results show that the use of SLU result can improve the search accuracy. Although the raw queries without SLU results could be used for the input of the vector space model, parsed queries shows better performances over different models. In addition, SLU results may be necessary to manage subsequent dialog, such as confirming the slot values and narrowing the candidate items. Next, HVSM shows the best performance on parsed and total queries. This means that slot information in MVSM may be useful to search more precisely in an actual system and that SVSM, not considering slot names, may be a necessary adjunct to overcome SLU errors. Finally, the results also show the back-off scheme is effective on unparsed inputs since it can return the relevant items even if the current query was OOG.

Query Type	SVSM (naïve)		SVSM (parsed)		MVSM		HVSM	
	P@100	*MRR*	*P*@100	*MRR*	*P*@100	*MRR*	*P*@100	*MRR*
Parsed	0.8410	0.5526	0.8619	0.6519	0.8033	0.6211	0.8954	0.7037
Unparsed	0.8152	0.5694	-	-	-	-	-	-
Total	0.8217	0.5652	0.8270	0.5902	0.8122	0.5802	0.8354	0.6022

Table 4 Evaluation results on noisy queries (WER=23.94%). SVSM (naïve) refers to SVSM with the whole query.

5.3 Evaluation on Noisy Queries

We generated noisy queries by using a simple ASR error simulator [1] applied to the textual queries. These are not real spoken queries, but artificially simulated queries given both a specific word error rate (WER) and error type distribution. We preliminarily evaluated the search performance on these queries (WER=23.94%) although simulated queries may differ from real queries produced in real ASR (Table 4). Out of 948 test queries, 239 queries had parse results and the F_1 score on parsed queries is 64.30%. The decrease in the ratio of parsed queries had an adverse effect in the performance of the proposed method. However, the HVSM still shows the best performance.

6 Conclusion and Discussion

We propose a hybrid approach to voice search using an effective vector space model, HVSM. HVSM provides the best performance on natural queries sourced through MTurk. This approach can consider the slot information and overcome OOG problems.

Some issues have yet to be resolved. The main issue is evaluating the book search model on spoken queries and not typed queries. We have collected speech data as spoken queries using MTurk and are investigating various ASR hypothesis structures (e.g. n-best list) as well as confidence scores that might be incorporated into our proposed model to realize robustness.

References

[1] Jung, S., Lee, C., Kim, K., Lee, G.G.: Data-driven user simulation for automated evaluation of spoken dialog systems. Computer Speech and Language **23**(4), 479–509 (2009)

[2] Marge, M., Banergee, S., Rudnicky, A.I.: Using the amazon mechanical turk for transcription of spoken language. In: Proc. ICASSP, pp. 5270–5273 (2010)

[3] Passonneau, R.J., Epstein, S.L., Ligorio, T., Gordon, J.B., Bhutada, P.: Learning about voice search for spoken dialogue systems. In: Proc. NAACL, pp. 840–848 (2010)
[4] Song, Y.I., Wang, Y.Y., Ju, Y.C., Seltzer, M., Tashev, I., Acero, A.: Voice search of structured media data. In: Proc. IEEE ICASSP, pp. 3941–3944 (2009)
[5] Voorhees, E.M., Tice, D.M.: The trec-8 question answering track evaluation. In: Proc. Text Retrieval Conference TREC-8, pp. 83–105 (1999)
[6] Wang, Y.Y., D.Yu, Ju, Y.C., Acero, A.: An introduction to voice search. IEEE Signal Processing Magazine **25**(3), 29–38 (2008)
[7] Ward, W., Issar, S.: Recent improvements in the cmu spoken language understanding system. In: Proc. ARPA Human Language Technology workshop, pp. 213–216 (1994)
[8] Yu, D., Ju, Y.C., Wang, Y.Y., Zweig, G., Acero, A.: Automated directory assistance system. In: Proc. INTERSPEECH, pp. 2709–2712 (2007)
[9] Zweig, G., Nguyen, P., Ju, Y.C., Wang, Y.Y., Yu, D., Acero, A.: The voice-rate dialog system for consumer ratings. In: Proc. INTERSPEECH, pp. 2713–2716 (2007)