Web-enhanced Content Retrieval for Information Access Dialogue System

Donghyeon Lee[1], *Cheongjae Lee*[2], *Minwoo Jeong*[1], *Kyungduk Kim*[1], *Seokhwan Kim*[1], *Junhwi Choi*[1] *and Gary Geunbae Lee*[1]

[1] Department of Computer Science and Engineering, POSTECH, South Korea
[2] Academic Center for Computing and Media Studies, Kyoto University, Japan
{semko, lcj80, stardust, getta, megaup, chasunee, gblee}@postech.ac.kr

Abstract

We consider the problem of content retrieval with complex queries for an information access dialogue system. Traditional information access dialogue systems rely on exact query matching and heuristic rules to find relevant content in a relational database. To deal with complex queries, a dialogue system is used to attain deep semantic processing such as full semantic parsing and ontology-based reasoning. However, these systems require a large amount of semantic annotation and domain expert knowledge that are often very expensive to obtain and thus have been limited in practice. In this paper, we present a simple alternative method where web-searched documents can contribute to enhanced vector space model-based content retrieval. Our model captures underlying co-occurrence patterns between the query and the contents. An efficient ranking algorithm is applied to retrieve the relevant contents. One merit of the proposed approach is that it does not require heavy semantic processing, and therefore, it results in efficient content retrieval. We demonstrate that our method is beneficial in an electronic program-guided dialogue system.

Index Terms: web-enhanced content retrieval, information access dialogue system

1. Introduction

In recent years, information access dialogue systems have been studied to construct a user-friendly interface to retrieve desired information. For example, a user can query the dialogue system to demand air flight information [1] and to search books [2] or electronic programs [3]. Traditional dialogue systems for information access store the *contents* of domain-specific information in a relational database (RDB), consisting of relational *tuples*. Content retrieval is a key component of information access dialogue systems to retrieve the relevant contents (i.e., a tuple in the RDB) for answering a user's request.

In most RDB-based dialogue systems, content retrieval is carried out as exact query matching and matching with heuristic query expansion rules. However, these systems often fail to deal with complex queries. As an example, consider a user query to an electronic program guide (EPG) system: "I want to watch Jisung Park's game." In practice, shallow semantic parsing is applied to extract reliable semantic information, e.g., `search_program(person=`'Jisung Park'`)`, and the dialogue system retrieves tuples with fields that match the query exactly or partially. As the contents database contains only a limited number of structured fields, such as program title and broadcast hours, the system returns no result for this example.

To overcome such a limitation, content retrieval is used to attain deep semantic processing, for instance, full semantic parsing [4] and ontology-based reasoning [5]. To make the complex query understandable, a full semantic parser might convert the raw sentence into first-order logic, such as `λx.game(x)∧roster(x, Jisung Park)`. Alternatively, a domain-specific ontology can be used instead of an RDB to perform reasoning on complex user queries [5]. Despite their generality and expressive power, both approaches have their own challenges: semantic parse tree-annotated corpora and ontology resources are scarce and difficult to obtain. In addition, semantic inferences based on these approaches are often very expensive. Therefore, such approaches have been limited in practice for information access dialogue systems.

In this paper, we propose an alternative approach to content retrieval that makes it feasible for our dialogue system to deal with complex queries. Instead of using heavy semantic processing, we use a vector space model (VSM) that forms a query and a tuple term vector. Our model accommodates web documents that are searched for each user query and tuple in the contents database to enhance the VSM with a scarcity of common terms. For example, the web search result for the phrase "Jisung Park" would contain expanded but closely related terms, such as "Manchester United" and "Premier League soccer player." We expect that the term distribution of web-searched documents of the relevant tuple is close to that of the documents searched by the user query. We argue that web documents could be a valuable resource to capture distributional similarity between the query and the contents. As a ranking algorithm, we present an efficient ranking function that combines two VSMs. As our method requires no expensive semantic analysis and additional expert knowledge, it results in efficient context retrieval. We apply our method to the EPG dialogue system, in which a user can query in natural language to access broadcast information. In our experiment, we demonstrate that the resulting method achieves a mean reciprocal rank of 0.775, and this compares favorably with the 0.576 shown by the baseline model.

The remainder of this paper is structured as follows. In Section 2, we give background on content retrieval for information access dialogue systems. Section 3 presents our web-enhanced content retrieval model and Section 4 provides an empirical evaluation of the proposed method. In Section 5, we conclude with a discussion of future work.

2. Content Retrieval for Information Access Dialogue Systems

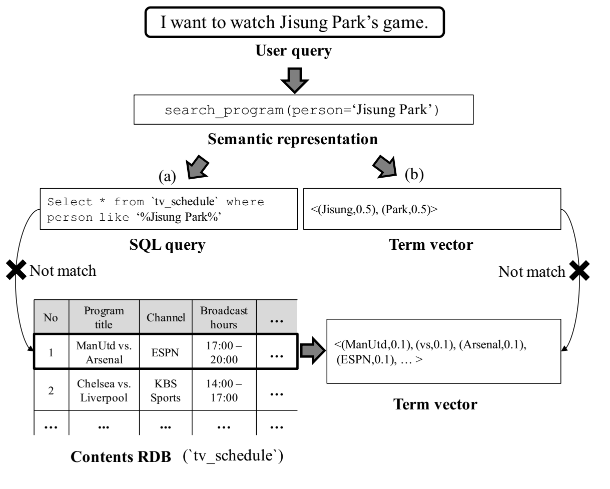
In this section, we present prior studies on content retrieval for information access dialogue systems and show challenges that make it possible to deal with complex user queries.

In information access dialogue systems that are implemented with an RDB as their knowledge content container, the goal of content retrieval is to find the most relevant tuple or a list of relevant tuples in the RDB tables, given a user query in natural language. As a natural language query is semantically ambiguous, the dialogue system first attempts to parse the speech for recognition. In practice, we are not always able to develop a full parser due to speech recognition errors and because of the characteristics of spoken language. Therefore, most dialogue systems apply domain-dependent, shallow semantic parsing to extract reliable semantic information (e.g., the user action and a domain-specific named entity) from the natural language query. Extracted information is structured as a *slot/filler frame*, and then a dialogue manager updates such a frame as a dialogue state with a discourse history. Traditional RDB-based dialogue systems implement exact matching rules of slot/filler frames, executing an SQL query to the contents of the RDB and using heuristic query expansion rules to achieve a better coverage. However, this method often fails to retrieve desired contents, in particular for *complex queries*. As shown in Figure 1(a), no match between the SQL queries and the RDB was made.

To alleviate this problem, there are roughly three categories of work: full semantic parsing, ontology-based dialogue systems and VSM-based content retrieval. In theory, full semantic parsing gives us a complete logical form for the underlying semantics. For example, for the phrase "Jisung Park's game," a user might mean "a list of all soccer games in which Jisung Park is rostered," with the first-order logic form `λx.game(x) ∧ roster(x, Jisung Park)`. In practice, however, this approach results in low accuracy when mapping ambiguous lexical items to semantic concepts (e.g., from a possessive, "Park's", to a `roster`). In addition, this approach needs a large number of semantic parse tree annotations to learn the semantic parser.

An ontology-based dialogue system was among the first applications of ontology-based reasoning to dialogue systems [5]. In [5], the investigators built a domain-specific ontology that consists of objects and their relations, representing a hierarchical structure of the semantic relatedness of concepts. For our example, "Jisung Park" is an instance of a `player` object that has an `isMemberOf` relationship to a `club` object. Although ontology-based reasoning is powerful, creating an ontology itself is challenging.

As an alternative to expensive semantic processing, we consider VSM-based content retrieval. In the context of information retrieval, each tuple in the RDB is treated as a (structured) document, and then a query-tuple VSM is used to yield a ranked list of tuples according to their relevance to the query. Some of the prior work on voice search presented VSM-based content retrieval in this manner [6, 7].


I want to watch Jisung Park's game.
User query
search_program(person='Jisung Park')
Semantic representation
(a)
(b)
Select * from `tv_schedule` where person like '%Jisung Park%'
SQL query
<(Jisung,0.5), (Park,0.5)>
Term vector
Not match
Not match
<(ManUtd,0.1), (vs,0.1), (Arsenal,0.1), (ESPN,0.1), … >
Term vector
Contents RDB (`tv_schedule`)

No	Program title	Channel	Broadcast hours	…
1	ManUtd vs. Arsenal	ESPN	17:00 – 20:00	…
2	Chelsea vs. Liverpool	KBS Sports	14:00 – 17:00	…
…	…	…	…	…

Figure 1: *An example of content retrieval for an information access dialogue system in the EPG domain.*

There has been only a limited amount of work applying VSM to content retrieval for information access dialogue systems, e.g. [8]. We construct a query term vector using the slot/filler frame (or natural language query) and a tuple term vector. Next, we assume that the distance between the query and the document vectors has a degree of relatedness. However, a complex query still remains a challenge. As shown in Figure 1(b), there is no overlapping term between the query vector and the tuple vector, so that the similarity score would be 0. This phenomenon occurs because both the query and the contents tuple tend to be too short to represent their meanings. Accordingly, we must bridge the gap between them. To address this problem, we will introduce a simple, but effective, method that uses web documents as additional knowledge.

3. Web-enhanced Content Retrieval

Our key idea for robust content retrieval is to use a collection of web-searched documents as an additional resource to capture underlying co-occurrence patterns between the query and the tuple. In this section, we implement our idea by extending the conventional VSM.

3.1. Model

Our content retrieval model consists of two VSMs: local and web-enhanced VSM. A local VSM (Local-VSM) forms a query vector from the slot/filler frame and a tuple vector from the tuple in the contents of the RDB. As both the frame and the tuple are structured, we can construct a structured term vector to represent structural information and can also consider a structural ranking algorithm, for example, BM25f [9]. As stated in [9], it is often expensive to tune the parameter (or weight) for each structured field. In this work, we assume an unstructured term vector that contains only lexical items in the frame and the tuple. As a term weight, we use a standard *tf.idf* scheme, in which we compute the term frequency and the inverse document (i.e., tuple) frequency. For example, Figure 1(b) shows a query and a tuple term vector for a Local-VSM. We omit the *idf* score and assume 10 terms in the first tuple for clarity. Unfortunately, Local-VSM does not make a significant value for a pair of vectors in our example, because there is no matched term.

A web-enhanced VSM (Web-VSM) is used to resolve this non-matching term problem. We begin by illustrating how the tuple term vector takes advantage of web documents. For each tuple in a database, we use attribute values (that is, lexical terms described above) as keyword phrases for the web search engine. Next, we construct a tuple vector that consists of a term weight computed over the obtained documents. To make our model more reliable and to accelerate ranking, we use a simple term selection technique; we exclude terms whose *idf* does not exceed a fixed threshold value (in our experiment, 0.3). We index all such web-enhanced tuple vectors in advance.

A query term vector in Web-VSM can be built in a similar way. In a preliminary experiment, we found, however, that the slot/filler frame is inaccurate when our shallow semantic parser takes a complex query. This result occurs because the complex query is more ambiguous. To avoid generating a deficient keyword phrase, we also consider all terms in the natural language query as a keyword phrase. For example, we use both "Jisung Park" and "I want to watch Jisung Park's game" as the keyword phrase. We construct two separate query term vectors using slot/filler and full sentence keyword phrases.

A premise in our web-enhanced content retrieval is that web documents are beneficial for accommodating expanded terms that may be closely related to terms appearing in the query and tuple. However, searching the web might increase the overhead needed to process all of the queries. To alleviate this problem, we use a two-step approach; we first do content retrieval with a Local-VSM; then, we search the web to compute a query vector for the Web-VSM when the previous retrieval stage produces no result or when all tuples do not exceed the threshold. Consequently, this process reduces the cost of searching the web, and Web-VSM focuses on complex queries that are not covered by Local-VSM. The overall process of our web-enhanced content retrieval is depicted in Figure 2.

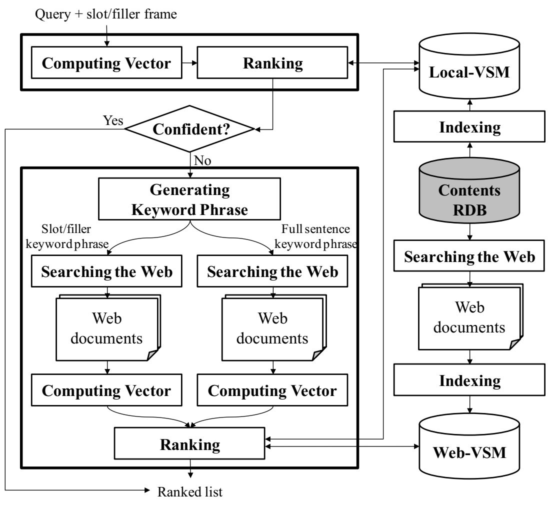


Figure 2: *A diagram that summarizes the overall process of web-enhanced content retrieval.*

3.2. Ranking

We formulate content retrieval as ranking the tuples with respect to the content of an RDB, given a user query. A merit of our approach is that any standard ranking algorithm is applicable in the framework once the query and tuple term vectors are computed. In this work, we adopt the cosine similarity function as our ranking basis because of its efficiency. We present a formal description of our ranking function for completeness.

Let us assume that we are given N tuples the content of an RDB. We define t_i as the i-th tuple ($1 \leq i \leq N$) and q as a user query. Next, we notate $\vec{t}_i^l$ as a tuple term vector generated for a Local-VSM and $\vec{t}_i^w$ as a web-enhanced tuple vector. As presented in Section 3.1, we define two query vectors by keyword phrases: $\vec{q}_1^l$ and $\vec{q}_1^w$ for the slot/filler keyword and $\vec{q}_1^l$ and $\vec{q}_2^w$ for the full sentence keyword. Finally, our ranking score function is defined as follows:

$$score(q, t_i) = \underbrace{\sum_{k\in\{1,2\}} \alpha_k \cdot f\left(\vec{q}_k^l, \vec{t}_i^l\right)}_{\text{Local-VSM}} + \underbrace{\sum_{k\in\{1,2\}} \beta_k \cdot f\left(\vec{q}_k^w, \vec{t}_i^w\right)}_{\text{Web-VSM}}$$

where α_1, α_2 and β_1, β_2 are weights, and f is a similarity function. We set the weights empirically as (α_1, α_2)=(0.5, 0.3) and (β_1, β_2)=(0.2, 0.1) to reflect the intuition that terms appear in the contents of the RDB with more confidence. Note that we use the Local-VSM part for the first step whereas the second step uses both parts to compute the ranking scores. The function $f(\vec{x}, \vec{y})$ can be any similarity function that reflects a degree of relatedness between two vectors, $\vec{x}$ and $\vec{y}$. This work focuses a simple cosine similarity function as $f(\vec{x}, \vec{y}) = (\vec{x} \cdot \vec{y})/\|\vec{x}\| \cdot \|\vec{y}\|$.

3.3. Relation to Other Methods

In this section, we give a qualitative analysis that illustrates how Web-VSM resolves complex queries and also compares our model with other approaches.

Our model accommodates the web documents to expand the term vectors. In the context of traditional information retrieval, the model is closely related to query expansion. Our work differs from traditional query expansion because we expand the tuple vector as well as the query vector. This expansion yields improved term vectors to supplement non-matching terms between the query and the tuple; for example, “Manchester United” and “football” are expanded common terms of q and t_1 that do not appear in both q and t_1.

The main advantage of the proposed method is that our model needs no domain-specific knowledge to represent semantic relatedness. In other approaches, such as full semantic parsing and ontology-based reasoning, explicit semantic analysis is a must, e.g., `game/roster` predicates, `player/club` objects and `isMemberOf` relationships should be explicitly modeled. Often, the explicit semantic representation requires expensive supervision. On the other hand, our work uses a counter approach that implicitly represents a degree of semantic relatedness between the query and the contents in a form of term distribution. Figure 3 demonstrates that the term distribution of semantically related tuples is close to that of the queries.

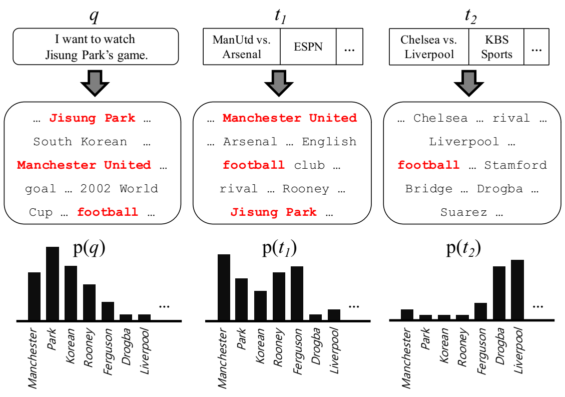


Figure 3: *Term distributions of query and tuples. Tuple t_1 is a relevant tuple of query q, but t_2 is not.*

4. Experiments

4.1. Experimental Setting

We applied our method to a Korean EPG dialogue system, in which the contents database consists of 1023 tuples, each of which is a unique TV program scheduled for broadcasting in Korea. We collected the database over the period of a week (January 17–23, 2011). We integrated our VSM-based content retrieval system to the dialogue system implemented in [3]. To obtain the web search results page, we used the API provided by Naver[1]. After inspecting the contents database, we found that some tuples have values that consist of only stop words, e.g., “with you.” Thus, we did not remove stop words because some stop words in Korean are necessary for our purpose.

Our goal is to evaluate how content retrieval outputs properly. Thus, we evaluated our method on 200 queries that were collected manually from Smart-TV users. In traditional

[1] http://dev.naver.com/openapi/

dialogue systems, the exact matching method could not handle 104 queries. This proportion verifies why robust content retrieval is needed for the EPG domain dialogue system.

To evaluate the relevance of a ranked list of tuples produced by our model, we used two standard metrics in the information retrieval task. Precision at *n* (*P@n*) represents the number of correct queries among the top *n* relevant lists divided by the total number of queries. We also present the mean reciprocal rank (MRR) in our result, which is the average of the reciprocal ranks of results for a sample of queries. For both metrics, a higher value is better. While there might be multiple correct answers, we assume a single correct tuple for a user query for brevity.

We also conducted a simple user test to identify the user preference. For this test, we compared Web-VSM with Local-VSM. An evaluation tool shows two results in random order for when the user marks his/her preference between Local-VSM and Web-VSM results. A total of 10 testers participated in this experiment.

4.2. Results

Table 1 summarizes the results that were obtained. We set 'Local-VSM', described in Section 3, as our baseline. Also, we considered two keyword phrases independently: using the query vector generated by a slot/filler keyword (Key1) and by the full sentence keyword (Key2). Both models substantially outperform the baseline (the second and third line in Table 1). The difference is statistically significant with the level $p < 0.05$ measured with the paired t-test. The significant improvement over the Local-VSM results demonstrates the benefits of using web documents for content retrieval with complex queries. Interestingly, Key1 and Key2 results are not significantly different, but additional benefits are obtained by combining the two (the last line in Table 1). This result is significantly better than both the Key1 and Key2 results ($p < 0.05$).

To inspect the effects of expanding tuple vectors, we consider a variant of Web-VSM where we did not use web documents to compute the expanded tuple term vector; that is, the ranking score solely depended on $\vec{t}_i^l$ but not on $\vec{t}_i^w$. We achieved an MRR score of 0.653, 0.656 and 0.688 for Key1, Key2, and Key1+Key2 schemes, respectively. This result shows that lexical terms in the contents database are also insufficient to express the degree of semantic relatedness.

To better understand the difference between Local-VSM and Web-VSM, we now turn to a user test evaluation. In this experiment, a query and two top 3 lists of system A and B (A and B are random) are shown on the screen. Then, a user can be assessed with 4 choices: A is preferred, B is preferred, both are acceptable, and both are unacceptable. Figure 4 summarizes the results of the preference test. In the figure, the user favors the Web-VSM result three times as often as that of the Local-VSM. Conversely, both systems incorrectly retrieve the relevant contents for approximately 12% of the queries. Because most of the errors are due to unreliable web documents, we may need a method to select the confident web pages. We leave this task to future work.

Table 1. *Result on content retrieval for the EPG dialogue system.*

	P@1	P@5	P@10	MRR
Local-VSM	0.540	0.625	0.640	0.576
+Web-VSM (Key1)	0.695	0.785	0.810	0.739
+Web-VSM (Key2)	0.685	0.770	0.800	0.727
+Web-VSM (Key1+2)	**0.730**	**0.820**	**0.845**	**0.775**

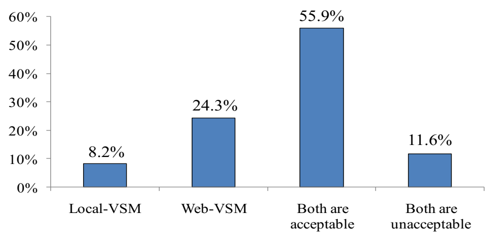


Figure 4: *Result of user preference on 200 test queries.*

5. Conclusions

In this paper, we presented web-enhanced content retrieval for information access dialogue systems. We demonstrated that our method is beneficial for running complex queries. Our model can capture term co-occurrence patterns to express the degree of semantic relatedness in a simple way. Without any semantic analysis and additional knowledge sources, our approach achieved a significant improvement over the baseline in the EPG dialogue system.

In future research, we plan to investigate a model that explicitly uses discourse history and updates dialogue states with a web-enhanced retrieval model. This approach may need to perform a mapping of topical terms to the shallow semantics defined by the slot/filler frame. Another interesting direction would be to perform integration of advanced information retrieval techniques, such as topic modeling in the content database, to improve ranking.

6. Acknowledgements

This research was supported by the MKE (The Ministry of Knowledge Economy), Korea, under the ITRC (Information Technology Research Center) support program supervised by the NIPA (National IT Industry Promotion Agency) (NIPA-2011-C1090-1131-0009), and the Intelligent Robotics Development Program, one of the 21st Century Frontier R&D Programs funded by the Ministry of Knowledge Economy (MKE).

7. References

[1] E. Levin, S. Narayanan, R. Pieraccini, K. Biatov, E. Bocchieri, G. Di Fabbrizio, W. Eckert, S. Lee, A. Pokrovsky, M. Rahim, P. Ruscitti, and M. Walker, "The AT&T-DARPA Communicator Mixed-Initiative Spoken Dialogue System," in Proc. ICSLP, 2000.
[2] C. Lee, A. Rudnicky, G. G. Lee, "Let's Buy Books: Finding Ebooks using Voice Search", In Proc. IEEE workshop on SLT, 2010.
[3] C. Lee, S. Jung, S. Kim, G. G. Lee, "Example-based Dialog Modeling for Practical Multi-domain Dialog System", Speech Communication 51(5), 466-484, 2009.
[4] L. Zettlemoyer and M. Collins. "Online Learning of Relaxed CCG Grammars for Parsing to Logical Form," In EMNLP-CoNLL, 2007.
[5] H. Noh, C. Lee, and G. G. Lee, "Ontology-based Inference for Information-seeking in Natural Language Dialog System", Proceeding of the 6pth IEEE INDIN, 2008.
[6] D. Yu, Y.-C. Ju, Y.-Y. Wang, G. Zweig, and A. Acero, "Automated Directory Assistance System", In Proc. INTERSPEECH, 2007.
[7] M. L. Seltzer, Y Ju, I. Tashev, and A. Acero, "Robust Location Understanding in Spoken Dialog Systems using Intersections", In Proc. INTERSPEECH, 2007.
[8] C. González-Ferreras, V. Cardeñoso-Payo, "Development and Evaluation of a Spoken Dialog System to Access a Newspaper Web Site," In Proc. INTERSPEECH, 2005.
[9] S. Robertson, H. Zaragoza, and M. Taylor. "Simple BM25 extension to multiple weighted fields," In ACM CIKM, pp. 42-49, 2004.