DYNAMICALLY SUPPORTING UNEXPLORED DOMAINS IN CONVERSATIONAL INTERACTIONS BY ENRICHING SEMANTICS WITH NEURAL WORD EMBEDDINGS

Yun-Nung Chen and Alexander I. Rudnicky

School of Computer Science, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213-3891, USA
{yvchen, air}@cs.cmu.edu

ABSTRACT

Spoken language interfaces are being incorporated into various devices (e.g. smart-phones, smart TVs, etc). However, current technology typically limits conversational interactions to a few narrow predefined domains/topics. For example, dialogue systems for smartphone operation fail to respond when users ask for functions not supported by currently installed applications. We propose to dynamically add application-based domains according to users' requests by using descriptions of applications as a retrieval cue to find relevant applications. The approach uses structured knowledge resources (e.g. Freebase, Wikipedia, FrameNet) to induce types of slots for generating semantic seeds, and enriches the semantics of spoken queries with neural word embeddings, where semantically related concepts can be additionally included for acquiring knowledge that does not exist in the predefined domains. The system can then retrieve relevant applications or dynamically suggest users install applications that support unexplored domains. We find that vendor descriptions provide a reliable source of information for this purpose.

Index Terms— spoken language understanding (SLU), spoken dialogue system (SDS), distributional semantics, word embeddings.

1. INTRODUCTION

Spoken dialogue systems (SDS) are appearing on smart-phones and allow users to launch applications via spontaneous speech. Typically, an SDS needs predefined task domains to support the corresponding functions, such as "*setting an alert clock*" and "*query some words via a browser*". However, the SDS is unable to dynamically support functions provided by newly installed or not yet installed applications. We address the following question: with an open domain spoken query, how can we dynamically and effectively provide the corresponding functions to fulfill users' requests? In this paper we present an approach to understanding a user's query and identifying the applications that can support such open domain requests.

Spoken language understanding (SLU) is important for building a good SDS, and slot filling and semantic parsing can be used to detect concepts for better understanding users' utterances. In recent work, Chen *et al.* explored a linguistically principled theory for spoken language, and demonstrated that frame-semantic parsing can be applied to slot induction in an unsupervised fashion, potentially alleviating labor cost of designing an SDS [1, 2]. However, some slots corresponding to domain-specific named entities cannot be covered by the generic semantic parser, while these slots may carry important semantics of the utterances. Hence, structured knowledge has been applied to capture domain-specific information and improved many applications in natural language processing [3, 4, 5]. Recently reported work has described approaches that leverage external semantic resources for unsupervised SLU for SDS [6, 7, 8, 9]. The knowledge graph was used to train models for intent detection in SLU, and results obtained from an unsupervised training process aligned well with the performance of traditional supervised learning [6]. Tur *et al.* also showed that search engine logs and entity types from the knowledge graph can infer implicit semantics and help improve the slot-filling performance in a movie domain [7, 10]. Such knowledge can be applied to domain expansion in SDSs [8]. On the other hand, the distributional view of semantics hypothesizes that words occurring in the same contexts may have similar meanings [11, 12]. With the recent advance of deep learning techniques, the continuous-valued word embeddings have further boosted the state-of-the-art results in many applications [13, 14, 15, 16].

This paper presents an approach that leverages the recent success of deep learning—we use the continuous-valued word embeddings to involve semantic representations in task-oriented SDSs. Here we use frame-semantic parsing and entity linking methods based on structured knowledge resources, which are to locate the slot fillers in a given query, and then the types of identified fillers are extracted as semantic seeds of the query. In order to understand users' requests, the semantic seeds are used to obtain semantically related knowledge via neural word embeddings. This enables an SDS to dynamically support non-predefined domains based on the semantics-enriched query. We evaluate the performance by examining whether retrieved applications can fulfill users' requests. Preliminary results show that our approach can allow an SDS to provide better responses when processing requests referring to non-predefined domains.

2. PROPOSED FRAMEWORK

Under the application-oriented SDS, the main idea is to predict users' intents, and then dynamically support functions corresponding to their requests, which do not need to focus on predefined domains. Therefore, it can be improved by providing more flexible communication after overcoming domain restrictions. Here we formulate the task as a ranking problem: given an user's spoken utterance, how can the system retrieve the relevant applications for supporting user's desired function in an unsupervised way?

This paper proposes to first identify the slot-fillers and enrich the semantics with mined domain knowledge. The whole system framework is shown in Fig. 1. In the first stage, we perform frame-semantic parsing and entity linking of structured knowledge elements, Wikipedia pages and Freebase nodes, and extract types of slots as semantic seeds [17, 4], which is introduced in Section 3. The second stage enriches the semantics using neural word embeddings to expand domain knowledge, where the detail is presented

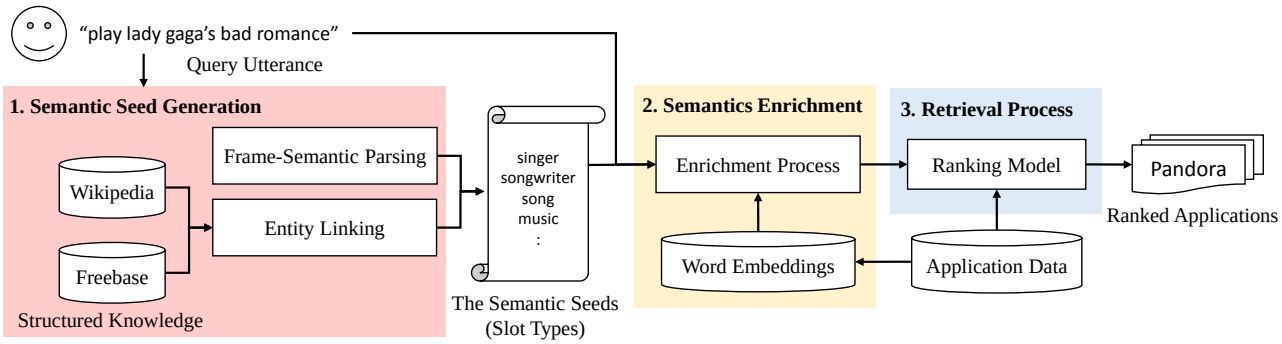

Fig. 1. Proposed framework.

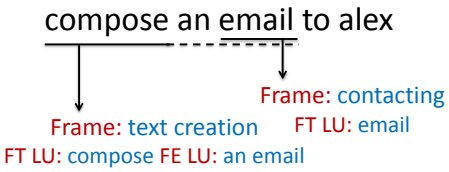

Fig. 2. An example of frame-semantic parsing on a query. FT: frame target. FE: frame element. LU: lexical unit.

in Section 4. By leveraging the semantic resources, we reformulate the query to retrieve domain-related information, and our retrieval model is performed to provide the ranking list of applications for users, which is described in Section 5.

3. SEMANTIC SEED GENERATION

Considering that types of slots may imply semantic meaning of the utterance, such information is mined to form the semantic seeds for expanding domain knowledge. Below we introduce two generation approaches, where one uses semantic parsing to identify semantically important concepts from outputted frames, and another uses entity linking of structured knowledge elements to mine entity types.

3.1. Frame Type of Semantic Parsing

FrameNet is a linguistically-principled semantic resource that includes considerable annotations about predicate-argument semantics, and its associated lexical units in English [18], developed based on Frame Semantics [19]. The theory believes that the meaning of most words can be expressed on the basis of semantic frames, represented as three major components: *frame (F), frame elements (FE)*, and *lexical units (LU)*. SEMAFOR[1] is a state-of-the-art semantic parser trained on FrameNet [17]. Fig 2 is an example of an ASR-decoded utterance parsed by SEMAFOR. The outputted frames carry important semantic concepts and can be viewed as slots of SDS [1].

In our approach, we parse all utterances using SEMAFOR, and extract all outputted frames. The words corresponding to outputted frames can be viewed as slot-fillers, which may contain important domain concepts of the utterances. To support open domain requests, the types of slots are extracted as semantic seeds to expand domain knowledge. For instance, we assume that the system does not cover the email writing domain and an user makes a request "*compose an email to alex*" to the system. Then the utterance can be parsed into two frames shown in Fig. 2, (text creation and contacting), and the outputted slot-fillers ("*compose*" and "*email*") corresponds to parsed frame targets. The types of slots "*text creation*" and "*contacting*" can be viewed as domain knowledge of the query, and then we use them as semantic seeds to expand domain-related information for better understanding. Given a query $Q = x_1, ..., x_i, ..., x_{|Q|}$, where x_i is the i-th word in the query, we define a set of semantic seeds $S_{\text{frm}}(Q) = \{s \mid \text{the frame labelled as } s \text{ has the slot filler } x_i \in Q\}$, where a semantic seed $s \in S_{\text{frm}}(Q)$ refers to a slot type outputted by SEMAFOR. The above example generates the semantic seeds, $s_1 =$ "*text creation*" and $s_2 =$ "*contacting*".

3.2. Entity Type from Linked Structured Knowledge

Considering that semantic parsing cannot recognize domain-specific named entities, some slot fillers would be missing and then some important domain knowledge cannot be included by $S_{\text{frm}}(Q)$. For instance, the utterance "*play lady gaga's bad romance*" asks for the applications with functionality about music playing, but frame-semantic parsing cannot accurately identify the slot about music based on the utterance. Hence, we first detect all entity mention candidates in the given utterance, which may serve as keywords inside the application, and use entity linking to mine entity types as semantic seeds. In the example above, the slot fillers include the singer "*lady gaga*" and her song "*bad romance*". The types of the slots, "*singer*" and "*song*", also viewed as entity types, provide domain-related cues and may effectively retrieve the applications about music playing. Below we propose to utilize external structured knowledge resources to mine entity types.

To identify the entities, we first parse the utterances by Stanford Parser[2] and capitalize all words in noun phrases. Then entity mention candidates are formed by the consecutive capitalized word chucks, and we use regular expressions to match the longest surface forms for refining them, considering that entities sometimes overlap with each other [13, 4]. The formed entity mention candidates may include longer pattern such as "*the trailer of iron man three*".

Given a query utterance $Q = x_1, ..., x_{|Q|}$ with entity mention candidates $M(Q) = \{m_1, ..., m_N\}$, where m_i is a word segment including consecutive words in Q, we propose to generate a sets of linked elements $L(Q) = \{l_1, ..., l_N\}$ and an associated semantic seed set $S(Q) = \{s_1, ..., s_N\}$ by two knowledge resources. The linked elements are Wikipedia pages and Freebase node lists presented in Section 3.2.1 and Section 3.2.2 respectively.

[1] https://code.google.com/p/semafor-semantic-parser/

[2] http://nlp.stanford.edu/software/

3.2.1. Wikipedia Page Linking

With the entity mention set for the query Q, $M(Q)$, we output $L_{\text{wk}}(Q) = \{l_1, ..., l_N\}$ as a set of linked Wikipedia pages ($L(Q)$ described above) with corresponding linking weights. Here the procedure is implemented by Illinois Wikifier[3] using an Integer Linear Programming (ILP) formulation to generate the mapping from mentions to Wikipedia pages [3, 4]. Then the associated semantic seed set, $S_{\text{wk}}(Q) = \{s_1, ..., s_N\}$, is formed, where a semantic seed $s_i \in S_{\text{wk}}(Q)$ is extracted from its linked Wikipedia page l_i and refers to the type of the mention m_i.

We observe that the first sentences in the Wikipedia pages usually define the entities; for example, the first sentence of linked Wikipedia page about the entity "*lady gaga*" is "*Stefani Joanne Angelina Germanotta, better known by her stage name Lady Gaga, is an American singer and songwriter.*" This includes the entity types, "*American singer*" and "*songwriter*". Hence, the first sentence in the Wikipedia page l_i is used to extract the semantic seed s_i for the mention m_i, where s_i includes all words parsed into adjectives or nouns in the noun phrase just following the part-of-speech pattern (VBZ) (DT) such as "*is a/an/the*". The semantic seed extracted from the sentence defining the entity "*lady gaga*" would be "*American singer and songwriter*". The generated semantic seed set $S_{\text{wk}}(Q)$ is refined by filtering out the function words, and s_i is set to be empty if the linking weight is lower than a threshold in order to eliminate unreliable semantics.

3.2.2. Freebase List Linking

Similarly, we propose to extract the semantic seeds that indicate entity types from another well-structured knowledge resource, Freebase[4]. Each mention $m_i \in M(Q)$ can link to a ranked list of Freebase nodes by Freebase API[5], and we define $L_{\text{fb}}(Q) = \{l_i \mid l_i\text{is a ranked list of Freebase nodes corresponding to } m_i\}$. We extract the top K notable types for each l_i as the semantic seed s_i to form $S_{\text{fb}}(Q)$.

4. SEMANTICS ENRICHMENT

Considering to include open domain knowledge based on the user's utterance, with the semantic seeds generated from the first stage, we utilize distributed word representations to capture syntactic and semantic relationship for expanding the domain knowledge [20].

The word representations are learned from a recurrent neural network language model [21], which uses the history to include the long distance related information. The word relationships are present as vector offsets, where all pairs of words sharing a particular relation are related by the same constant offset in the embedded space. To involve distributional semantics, we transform each token x into its embedding vector $\mathbf{x}$ by pretrained word embeddings. Then the similarity between a pair of tokens x_a, x_b, $\text{Sim}(x_a, x_b)$, can be computed as their cosine similarity in the continuous vector space [20]. With embedding vector $\mathbf{x}$ corresponding to the token x, we build a vector $\mathbf{r}_x = [r_x(1), ..., r_x(t), ..., r_x(T)]$ to indicate semantically related words, where T is the vocabulary size of embedding training data. There are two proposed methods of computing $\mathbf{r}_x$, where we use notations $\mathbf{r}_x^1$ and $\mathbf{r}_x^2$ referring to $\mathbf{r}_x$ derived from two methods below.

- K Nearest Neighbors (KNN)

$$r_x^1(t) = \begin{cases} 1 & \text{, if } y_t \text{ is the word whose embedding vector has top } K \text{ greatest similarity to } \mathbf{x}. \\ 0 & \text{, otherwise.} \end{cases} \tag{1}$$

- Threshold Filtering

$$r_x^2(t) = \begin{cases} 1 & \text{, if } \text{Sim}(\mathbf{y_t}, \mathbf{x}) \geq \delta. \\ 0 & \text{, otherwise.} \end{cases} \tag{2}$$

The t-th element of vector $\mathbf{r}_x$ is decided based on the similarity between the embeddings of the token x and the token y_t (t-th token from the vocabulary). $\mathbf{r}_x$ can be viewed as a vector indicating the nearest neighbors of the token x in the continuous vector space by restricting the number of neighbors (K) or the smallest similarity (δ). Note that $r_x(t) = 1$ when $x = y_t$, which implies that the nearest neighbors of the token x include itself.

This procedure is to obtain the semantically related knowledge for improving the query. For example, "*compose an email to alex*" focuses on email writing domain, and the generated semantic seeds may include the tokens "*text*" and "*contacting*". Then word embeddings can help provide additional words with similar concepts. For example, the nearest vectors of "*text*" include "*message*", "*msg*", etc. in the continuous space. Section 5 describes how to leverage the information by reformulating the query to involve conceptually related knowledge during retrieval process, so that the system can provide proper applications for supporting open domain requests.

5. RETRIEVAL PROCESS

5.1. Query Reformulation

Given a spoken query Q, we build a corresponding query vector $\mathbf{q} = [q(1), ..., q(t), ..., q(T)]$ and $q(t)$ is equal to the frequency of the token y_t (t-th token of vocabulary) in Q. Then we propose two methods to construct the updated query vector $\mathbf{q}'$:

- Embedding-Enriched Query

$$\mathbf{q}' = \sum_{x \in Q} \mathbf{r}_x, \tag{3}$$

  which can be viewed as integration of the tokens that are semantically similar to any of words in the original query.

- Type-Embedding-Enriched Query

$$\mathbf{q}' = \sum_{x \in Q \cup S(Q)} \mathbf{r}_x, \tag{4}$$

  which uses the generated semantic seeds $S(Q)$ from Section 3 to additionally include domain-related knowledge.

With updated query vector $\mathbf{q}'$, we can have corresponding word query Q' and estimate its query likelihood in Section 5.2.

5.2. Ranking Model

As a task of ranking applications based on the user's spoken query, we use a language modeling approach for query likelihood estimation [22], and the applications are ranked by

$$P(Q \mid A) = \frac{1}{|Q'|} \sum_{x \in Q'} \log P(x \mid A), \tag{5}$$

[3] http://cogcomp.cs.illinois.edu/page/software_view/Wikifier

[4] https://www.freebase.com

[5] https://developers.google.com/freebase/

where Q' is the query Q after reformulation, A is a considered application, x represents the token in the query, and $P(Q \mid A)$ represents the probability that user speaks Q to make the request for launching the application A. For example, in order to use the application "`Gmail`", an user is more likely to say "*compose an email to alex*", while the same utterance should correspond to a lower probability when launching the application "`Maps`". To estimate the likelihood by the language modeling approach, we define A as the description content of an application with assumption that it carries semantically related information. For example, the description of "`Gmail`" includes the text segment "*read and respond to your conversations*", and "`Maps`" includes texts "*navigating*" and "*spots*" in its description.

With the reformulated query, we rank applications in an unsupervised way. The target of ranking model is to help the SDS obtain the knowledge that may not be in defined domains. Assuming that the original system does not cover email writing domain, we hope to retrieve email-related applications for dynamically and effectively supporting the unexplored domain. From a practical perspective, systems can properly interact with users to fulfill their open domain requests, eliminating domain restriction.

6. EXPERIMENTAL SETUP

6.1. Domain Definition

Considering to expand useful domains of SDS, we extract the most popular applications from mobile app stores to define important domains users tend to access frequently, and the defined domains are used to design the experiments for the task. Fig. 3 shows total 13 domains we define for the experiments. The speech corpus used in this experiment is collected from 5 non-native subjects (1 female and 4 males). They are only provided with pictures referring to domain-specific tasks in a random order, and for each picture/task a subject is asked to use 3 different ways to make requests for fulfilling the task implied by the displayed picture. Thus 39 utterances (total 13 tasks and 3 ways for each) are collected from each subject. Fig. 3 shows provided pictures and implied tasks, and some recording examples are shown in Table 1. The corpus contains 195 utterances. An automatic speech recognition (ASR) system was used to transcribe speech into text, and the word error rate is reported as 19.8%. Here we use Google Speech API to perform better recognition results because it covers more named entities, which may be out-of-vocabulary words for most recognizers. The average number of words in an utterance is 6.8 for ASR outputs and 7.2 for manual transcripts, which implies the challenge of retrieving relevant applications with limited information in a query.

Table 1. The recording examples collected from some subjects.

Task ID	Transcript of the utterance
3	*please dial a phone call to alex* *can i have a telcon with alex*
10	*how can i go from my home to cmu* *i'd like navigation instruction from my home to cmu*

6.2. Application Data Collection

The data for retrieval archive was collected from Google Play[6] in November 2012. Each Android app in Google Play has its own description page, and the extracted metadata we use includes its name, number of downloads[7], and content description. The total number of considered applications is 140,854[8]. For evaluation, subjects are asked to manually annotate applications from Google Play that can support the corresponding tasks. Then we use the subject-labelled applications as our ground truth for evaluating our returned applications, where we use standard metrics of information retrieval, mean average precision (MAP) and precision at 5 (P@5).

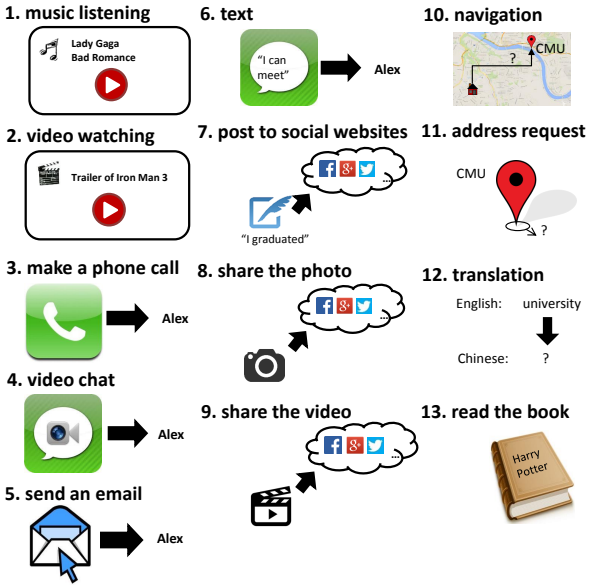

Fig. 3. Total 13 tasks in the corpus (only pictures are shown to subjects for making requests).

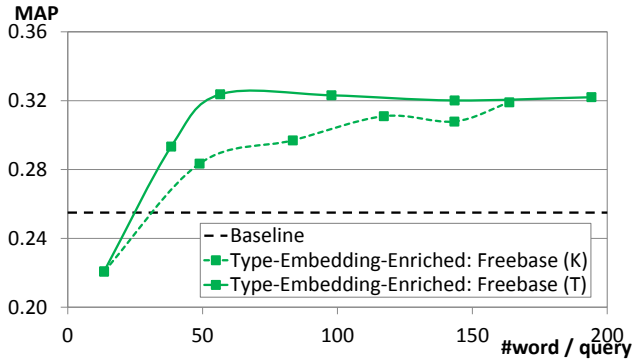

Fig. 4. MAP of results using entity types extracted from Freebase on ASR outputs (K: KNN, T: threshold filtering).

6.3. Word Embeddings Model

To include the distributional semantics view for SLU,we use description content of all applications to train word embeddings using bag-of-words architecture [23], which predicts the current word based on the context[9]. The resulting vectors have dimensionality 300, and vocabulary size is 8×10^5.

[6] `http://play.google.com`

[7] Google does not provide the absolute number of downloads. Instead, it discretizes this number into several ranges.

[8] Google defines two major categories for the programs, "game" and "application". This paper only uses apps with category "application".

[9] `https://code.google.com/p/word2vec/`

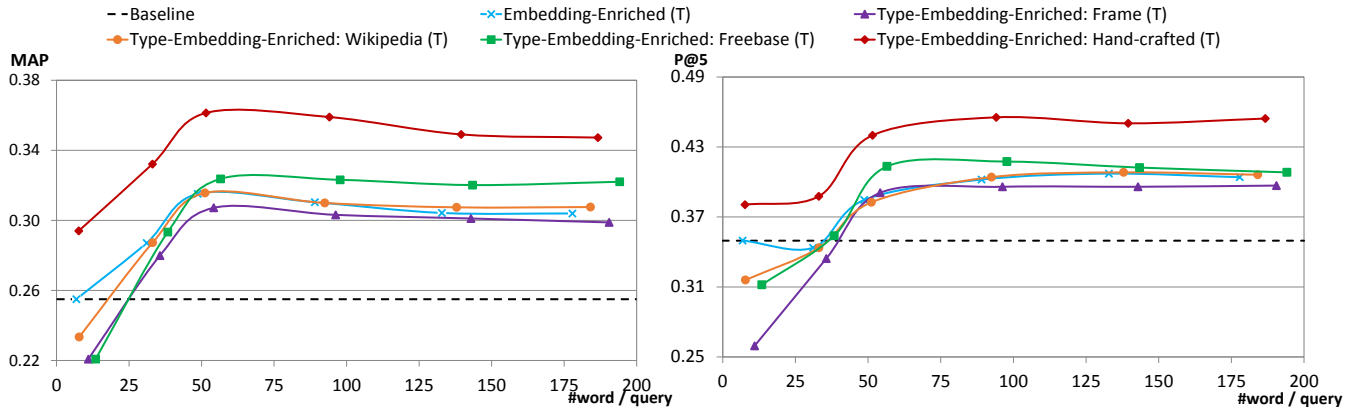

Fig. 5. MAP and P@5 of all results with Google Play word embeddings on ASR outputs (T: threshold filtering).

6.4. Retrieval Setup

Lemur toolkit[10] is used to perform our ranking model. For retrieval setting, word stemming[11] and stopword removal[12] are applied, and we assign equal weight to each term in the query to eliminate the influence of weighting [22]. To consider popularity of applications, for each returned list, we rerank the "popular" applications to the top of list, where "popular" means the applications with more than ten million downloads, because we assume that users are more willing to use/install popular applications, and also our ground truth is based on subjects' annotation, where most reference applications belong to the set of popular applications we define.

7. RESULTS AND DISCUSSIONS

Fig. 5 shows the results of our approach. Below we first compare the results between using KNN and threshold filtering, analyze the effectiveness of proposed approaches derived from different semantic seeds, and discuss overall results. Here the baseline uses the original query to retrieve relevant applications, and we show it in the figure as a reference.

7.1. Comparison between KNN and Threshold Filtering

With KNN or threshold filtering, we can increase K (number of neighbors) or decrease δ (threshold) to involve more domain-related knowledge. Since all results from two methods have similar trends, we only show the MAP of Type-Embedding-Enriched using Freebase with increasing average word counts per query from ASR outputs in Fig. 4 to analyze the difference. It is obvious that threshold filtering performs better than KNN, probably because threshold filtering method selects neighbors in the continuous space more precisely, while KNN may contain some words that are not semantically related, resulting in more noisy words in the reformulated query and decreasing the performance. Also, the threshold filtering method boosts the performance more efficiently compared to KNN, since the extracted information is more reliable.

7.2. Effectiveness of Semantics Enrichment

Fig. 5 shows the performance of proposed approaches using threshold filtering (threshold filtering performs better than KNN) with increasing average word counts per query from ASR outputs in terms of MAP and P@5. To validate whether the types of slots can help involve domain-specific knowledge, we use hand-crafted entity types in the experiments to show the upper bound of the performance. For example, we manually create the entity-type mappings from "*the trailer of iron man three*" to "*video*" in task 2, and from "*alex*" to "*contact*" in task 3-6.

In the beginning of semantics enrichment, the results of all approaches other than using hand-crafted mapping are worse than the baseline, but they are boosted very fast (large slope) when expanding more domain-related knowledge, and further beat the baseline. This means that the generated semantic seeds correctly expand domain-related words, and then the likelihood of relevant applications can be measured larger, resulting in better performance efficiently.

The embedding-enriched method (blue line marked with crosses) outperforms the baseline (dashed line without marks as a reference) even though it does not use any generated semantic seeds, showing that word embeddings can effectively enrich semantics only based on the original query. Among three structured knowledge resources in type-embedding-enriched methods, Freebase (green line marked with squares) performs best, because entities in Freebase have more precise types and then the semantic seeds can accurately obtain domain-related knowledge for improving performance. Also, the results of Freebase are better than the embedding-enriched method when query length is longer than about 50, especially for P@5, which means that we can effectively and efficiently expand domain-specific knowledge by using types of slots from Freebase, offering better responses in the uncovered domains. However, the method using Wikipedia (orange line marked with circles) relies on the fixed structure, which may not mine the correct and precise types of slots for obtaining domain knowledge. Also, the results using frame types as semantic seeds perform worst (purple line marked with triangles), which implies that for the application domain, users usually use many entities in their query that cannot be covered by the generic semantic parser, so that only including generic concepts is not enough to well understand users' requests. In the figure, hand-crafted mapping (red line marked with diamonds) significantly outperforms the baseline in the beginning and becomes better when applying semantic enrichment, showing that the correct types of slots can offer better understanding for retrieving relevant applications and also telling the room of improvement.

7.3. Overall Results

Table 2 shows the performance of all results with threshold filtering ($\delta = 0.35$) for ASR and manual transcripts. We find that applying neural word embeddings significantly improves performance for

[10] http://www.lemurproject.org/
[11] http://tartarus.org/martin/PorterStemmer/
[12] http://www.lextek.com/manuals/onix/stopwords1.html

Table 2. The performance with threshold filtering $\delta = 0.35$ (%).

Approach			ASR		Manual	
			MAP	P@5	MAP	P@5
Baseline: Original Query			25.50	34.97	26.76	35.90
Proposed	Embedding-Enriched		30.42	40.72	31.01	42.15
	Type-Embed.	Frame	30.11	39.59	31.05	41.03
		Wikipedia	30.74	40.82	31.59	**42.77**
		Freebase	**32.02**	**41.23**	**32.40**	42.36
		Hand-Crafted	34.91	45.03	35.81	47.08
Max Relative Improvement (%)			+25.5	+17.9	+21.1	+19.1

both ASR and manual transcripts. The type-embedding-enriched method with Freebase performs best for most results, achieving about 25% and 18% relative improvement for MAP and P@5 on ASR outputs respectively. The performance of type-embedding-enriched methods with hand-crafted mapping shows that there's a large room of improvement. In sum, this paper shows that word embeddings trained on vendor descriptions can effectively include semantically related information for retrieving relevant applications when considering types of slots as semantic seeds. Finally, the experiments demonstrate the possibility of supporting an open domain SDS in an unsupervised fashion.

8. CONCLUSION

This paper proposes an unsupervised approach for acquiring open domain knowledge based on the user utterances. Our work uses structured knowledge resources to extract types of slots as semantic seeds, and then leverages distributed word representations to obtain domain-related information. By using description of applications as a retrieval cue and reformulating the query with generated semantic seeds, which are related to uncovered domains, the system can retrieve more relevant applications in an unsupervised way. Hence, the proposed approach enables the system to properly react to users' queries, such as providing relevant applications or suggesting users install applications that support uncovered domains, to fulfill their open domain requests, providing more natural interactions and better user experiences in SDSs.

9. REFERENCES

[1] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky, "Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing," in *Proceedings of ASRU*, 2013.

[2] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky, "Leveraging frame semantics and distributional semantics for unsupervised semantic slot induction in spoken dialogue systems," in *Proceedings of SLT*, 2014.

[3] L. Ratinov, D. Roth, D. Downey, and M. Anderson, "Local and global algorithms for disambiguation to wikipedia," in *Proceedings of ACL*, 2011.

[4] X. Cheng and D. Roth, "Relational inference for wikification," in *EMNLP*, 2013.

[5] Ming-Wei Chang, Bo-June Hsu, Hao Ma, Ricky Loynd, and Kuansan Wang, "E2E: An end-to-end entity linking system for short and noisy text," *Making Sense of Microposts*, 2014.

[6] Larry Heck and Dilek Hakkani-Tür, "Exploiting the semantic web for unsupervised spoken language understanding," in *Proceedings of SLT*, 2012.

[7] Gokhan Tur, Minwoo Jeong, Ye-Yi Wang, Dilek Hakkani-Tür, and Larry P Heck, "Exploiting the semantic web for unsupervised natural language semantic parsing," in *Proceedings of INTERSPEECH*, 2012.

[8] Ali El-Kahky, Derek Liu, Ruhi Sarikaya, Gokhan Tur, Dilek Hakkani-Tür, and Larry Heck, "Extending domain coverage of language understanding systems via intent transfer between domains using knowledge graphs and search query click logs," in *Proceedings of ICASSP*, 2014.

[9] Yun-Nung Chen, Dilek Hakkani-Tür, and Gokhan Tur, "Deriving local relational surface forms from dependency-based entity embeddings for unsupervised spoken language understanding," in *Proceedings of SLT*, 2014.

[10] Dilek Hakkani-Tür, Asli Celikyilmaz, Larry Heck, Gokhan Tur, and Geoff Zweig, "Probabilistic enrichment of knowledge graph entities for relation detection in conversational understanding," in *Preceedings of INTERSPEECH*, 2014.

[11] Zellig S Harris, "Distributional structure," *Word*, 1954.

[12] Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman, "Indexing by latent semantic analysis," *JASIS*, vol. 41, no. 6, pp. 391–407, 1990.

[13] Richard Socher, Alex Perelygin, Jean Y Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts, "Recursive deep models for semantic compositionality over a sentiment treebank," in *Proceedings of EMNLP*, 2013.

[14] Tomáš Mikolov, *Statistical language models based on neural networks*, Ph.D. thesis, Brno University of Technology, 2012.

[15] Tomáš Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, "Efficient estimation of word representations in vector space," in *Proceedings of ICLR*, 2013.

[16] Wen-tau Yih, Xiaodong He, and Christopher Meek, "Semantic parsing for single-relation question answering," in *Proceedings of ACL*, 2014.

[17] Dipanjan Das, Desai Chen, André F. T. Martins, Nathan Schneider, and Noah A. Smith, "Frame-semantic parsing," *Computational Linguistics*, 2013.

[18] Collin F Baker, Charles J Fillmore, and John B Lowe, "The Berkeley FrameNet project," in *Proceedings of COLING*, 1998.

[19] Charles J Fillmore, "Frame semantics and the nature of language," *Annals of the NYAS*, vol. 280, no. 1, pp. 20–32, 1976.

[20] Tomáš Mikolov, Wen-Tau Yih, and Geoffrey Zwieg, "Linguistic regularities in continuous space word representations," in *Proceedings of NAACL-HLT*, 2013.

[21] Tomáš Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur, "Recurrent neural network based language model," in *Proceedings of INTERSPEECH*, 2010.

[22] Jay M Ponte and W Bruce Croft, "A language modeling approach to information retrieval," in *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, 1998.

[23] Tomáš Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean, "Distributed representations of words and phrases and their compositionality," in *Proceedings of NIPS*, 2013.