# An Intelligent Assistant for High-Level Task Understanding

**Ming Sun** **Yun-Nung Chen** **Alexander I. Rudnicky**
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
{mings, yvchen, air}@cs.cmu.edu

## ABSTRACT
People are able to interact with domain-specific intelligent assistants (IAs) and get help with tasks. But sometimes user goals are complex and may require interactions with multiple applications. However current IAs are limited to specific applications and users have to directly manage execution spanning multiple applications in order to engage in more complex activities. An ideal personal agent would be able to learn, over time, about tasks that span different resources. This paper addresses the problem of cross-domain task assistance in the context of spoken dialogue systems. We propose approaches to discover users' high-level intentions and using this information to assist users in their task. We collected real-life smartphone usage data from 14 participants and investigated how to extract high-level intents from users' descriptions of their activities. Our experiments show that understanding high-level tasks allows the agent to actively suggest apps relevant to pursuing particular user goals and reduce the cost of users' self-management.

### Author Keywords
Multi-domain; User intention; Spoken dialog system (SDS); Language understanding.

### ACM Classification Keywords
I.2.1 Applications and Expert Systems: Natural language interfaces; I.2.7 Natural Language Processing: Language Parsing and Understanding; I.2.11 Distributed Artificial Intelligence: Intelligent agents

## INTRODUCTION
Smart devices, such as phones or TVs, now host applications (apps) from different domains. Each app is designed to handle a limited number of domains (usually one). It may be configured by developers to support transition between known apps, but such functionality is not scalable and loses potentially desirable adaptive configuration. On the other hand, users can mentally arrange apps and seamlessly coordinate the information among them. However, this manual process of launching apps one by one may be time-consuming and difficult, especially for elder users and users with (visual) disabilities, although vocabularies of a touch-screen or gestures have been enriched significantly over the past decade [10]. We would want our personal IAs to help organize apps/domains automatically given user requests expressed at the level of intentions. For example, upon receiving "can you help me plan an evening out with my friends?", we would like our agent to find a restaurant with good reviews (YELP), reserve a table (OPENTABLE) and contact friends (MESSENGER).

Conventional multi-domain dialog systems passively select one domain from multiple domains according to a user input, ignoring relationships between domains and the ultimate user intention behind cross-domain behaviors [3, 4, 5, 6, 17, 22, 14, 21, 23, 13]. This paper describes a layer above individual applications, which links them to a specific intention underlying user activities [26, 27]. By doing so (and in combination with other techniques), an agent would be able to manage interactions at the level of intentions, mapping intents into multiple existing applications/functionality. In the example above, the agent may respond "Okay, to *plan a dinner event*, I need to know *where*, *when* and *who*". Here, "*plan a dinner event*" indicates a (in-)correct interpretation of the user intention. (The user would have opportunity to correct the agent when misunderstanding presents.) *Where*, *when* and *who* collectively construct a shared context across app boundaries. Thus, a unified interaction could be provided, instead of concatenating individual domains managed by the user. This paper focuses on an IA which is capable of 1) discovering meaningful intentions from user's past interactions; 2) leveraging surface intentions with groups of apps; 3) talking about intentions via natural language. In the rest of the paper, a real-life multi-domain dataset is briefly described, followed by our framework (HELPR). At the end, we discuss user studies that evaluate our model.

## DATA COLLECTION
We logged real-life interactions at app-level from users' smart phones. We then requested two types of user annotation: 1) what apps were used for a particualr goal; and 2) what the goal was (i.e., task description). Meta information such as date, time, location was shown to the user to aid recall. Users were also asked to re-enact the smart phone interaction by talking with a Wizard-of-Oz system. An example of annotation and Wizard-of-Oz dialog is shown in Figure 1.

We had 14 participants and collected 533 sessions; mean age for the 4 male participants was 23.0 and 34.6 for the 10 females. In the group were 12 native English speakers. Details of the collection are provided in [27].

## HELPR FRAMEWORK
The agent (see Figure 2) maintains an inventory of past interactions, such as "plan a trip to California", each associated


*IUI'16*, March 07–10, 2016, Sonoma, CA, USA.

ACM 978-1-4503-4137-0/16/03$15.00.
DOI: http://dx.doi.org/10.1145/2856767.2856818

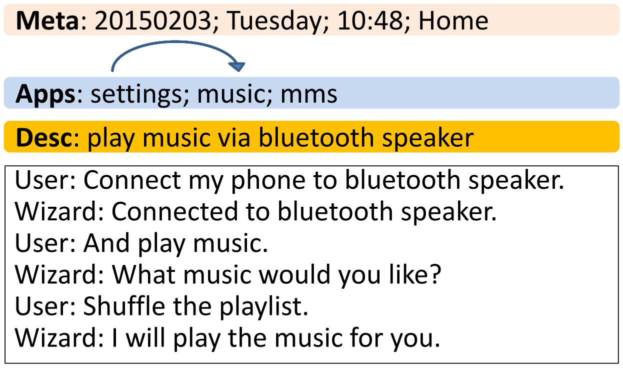


**Figure 1. User connected SETTINGS and MUSIC and noted that these two apps were used to *play music via bluetooth speaker*. Wizard-of-Oz dialog was collected and manually transcribed.**

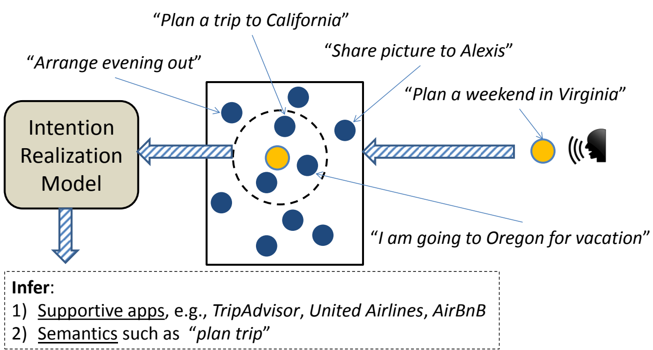


**Figure 2. Intention understanding and realization example. Solid nodes denote past interactions (blue) and current input (yellow).**

with information such as the sequence of apps used and the utterances spoken by user during the interaction. Given a new input (yellow node), the agent first identifies similar past experience (denoted as the ones within the dashed circle). This is the intention understanding process. Next, an intention realization model is built from those similar interactions to generate 1) supportive apps and 2) natural language reference. Thus, the intelligent agent transparently conveys its understanding of the input intention in these two modalities.

### Intention Understanding

We define a complex intention collectively by the set of previous interactions of similar nature. We used two approaches to find such similar past experience. A cluster-based method first groups training examples into $K_C$ clusters (i.e., segmenting the semantic space). The input language is used to identify the closest of these clusters; the members of this cluster define the nature of the intention. We also investigated a K-Nearest Neighbors approach that finds the $K_N$ most similar past interactions given the input.

We anticipate some major differences between cluster-based and neighbor-based intentions. (i) The cluster-based method should provide insight into the basic intentions of a user. This may become useful when the agent is actively learning tasks, i.e., asking user to label the current activity while suggesting one of the list of basic intentions. (ii) Cluster-based method can utilize richer contextual information (such as the apps used or utterances spoken by the user) when segmenting the semantic space of past interactions. Ideally, this yields better clustering performance. Such post-initiate information is not available in the neighbor-based approach, since it does not have a training process. (iii) The cluster-based approach has hard boundaries between intentions. Instances close to the boundaries may not be characterized well by their cluster members, compared with neighbor-based method. However, regardless of the differences between these two approaches, we believe that by referring to shared (past) experience, the agent can (i) better indicate (mis-)understanding of user's intention; and (ii) build rapport with the user [30].

### Intention Realization in Two Modes

#### *Supportive Applications*

As an intelligent user interface, the agent needs to assist human user in pursuing complex intentions that span multiple domains. We propose two strategies to generate sets of supportive apps. In the first one, we combine the individual app sequences in a set into a single app sequence to cover the activity (denoted as REPSEQ). For example, $\{a_1a_2a_3, a_1a_2a_4, a_1a_3a_4\}$ will yield $a_1a_2a_4$ by majority voting at each position. An alternate strategy would be to have a classifier assign multiple labels (apps ids) to the input (MULTLAB). The advantage of REPSEQ is that it can preserve common ordering among apps. However, from the example above, once the members are selected, the input language has no further influence on the selection of apps. Arguably, during this process we can weight each set member by its closeness to the input; we did not investigate this possibility. In this work, we focus on the quality of the proposed set of apps. At present we do not consider app order.

In the user interface, the agent could a) present the clickable icons of these apps to reduce the navigation through installed apps; b) warm up these apps to speed up the activation; c) build unified conversation based the set of apps.

#### *Language Reference*

The human-agent communication channel needs to be transparent in both directions. The agent must be able to verbally convey its understanding of user intention, allowing the user to track the agent's understanding. For example, it can use explicit or implicit confirmation [2], e.g., "do you want to *share a picture*?" Practically this can simply be a template ("do you want to ___?") and the reference to the intention ("share a picture"). Compared with echoing content extracted from the user's most recent input, our approach better communicates the agent's (mis-)understanding, allowing timely detection and recovery of errors.

To enable this we want our agent to automatically infer the semantics from related past experience. Text summarization can be used to generate a high-level description of the intention cluster [9, 15]. Keyphrase extraction provides an alternative [29, 18, 1]. In our case, we mainly need a short content ("share a picture") so the keyphrase approach is more suitable. Even if the automatic generation of semantic summarization is not precise, in context it may still be sufficiently meaningful to the human.

## USER STUDIES

### Study 1: End-to-End Evaluation

We investigated the differences within: 1) cluster-based vs. neighbor-based intention models; 2) personalized vs. generic setups; 3) REPSEQ vs. MULTLAB realization strategies. For each user, the chronologically first 70% of collected data was to train a personalized mode (in principle mirroring actual data accumulation). The remaining 13 users' first 70% data was combined and used to train a generic model. The number of intentions $K_C$ for the cluster-based intention model and the number of nearest neighbor $K_N$ for the neighbor-based model were tuned. $K_C$ was automatically optimized (from 1 to 10) via gap statistics [28]. $K_N$ was set to the square root of the number of training examples [7]. For REPSEQ we used ROVER [8] to collapse multiple app sequences into one. For MULTLAB, we used support vector machine (SVM) with linear kernel.

There are intra-user and inter-user inconsistencies in the use of language/apps, creating the problem of *vocabulary-mismatch* [16, 24], where interactions related to the same intention may have non-overlapping 1) spoken terms ("take picture" vs. "shoot photo"), sometimes caused by minor differences such as wrong word choice or morphology ("take" vs. "taking"); 2) app choice, e.g., people may use different apps with essentially similar purpose (MESSENGER vs. EMAIL). To address these issues, we applied query enrichment (QryEnr) and app similarity (AppSim). We describe them in detail.

#### *Query Enrichment*

QryEnr will expand the query by incorporating words semantically close to it [25], for example $\{shoot, photo\} \rightarrow \{shoot, take, photo, picture, selfie\}$. The chance of observing sparse input feature vectors caused by out-of-vocabulary (OOV) is thereby reduced. In this work, we used `word2vec` with the gensim toolkit[1] on the model[2] pre-trained on GoogleNews [20]. Each word $w_t$ in the pre-processed (lemmatization on verbs and nouns) query $Q = \{w_1, w_2, ..., w_T\}$ yields mass increases for $N$ semantically close words in the feature vector $f$ [27].

#### *App Similarity*

In the generic model, a recommended app, e.g., BROWSER may not match the only (or preferred) app on a specific user's phone, e.g., CHROME. Therefore, similarity metrics among apps are also needed to convert all apps in the generic model training data into the ones that are in this user's phone (as a pre-process). The other is to convert the recommendation results to fit this user's installed apps (a post-process). In the real world, pre-process may not be feasible since there are many individual users and adapting the (huge) generic training data for each of the users is expensive. Therefore, in this work we adopted post-processing.

We can construct a similarity matrix among all 132 apps in our collection by three means: (i) rule-based: the app package names can be useful, e.g., com.lge.music is close to com.sec.android.app.music since both contain the string "music"; (ii) knowledge-based: the Google Play store provides a finite ranked list of "similar apps" for each entry; (iii) data-based: app descriptions from the store can be projected into a high-dimensional semantic space to compute similarity. In the rule-based method, we used edit distance with 50 hand-crafted fillers (e.g., "com", "android") removed from package names. For the knowledge-based approach, we used reversed rank $(1/r)$ as the similarity. For the data-based approach, we used the `doc2vec` toolkit to train the space for over 1 million apps then used cosine similarity [12]. Knowledge-based and data-based matrices are sparse since some (vendor) apps were not found in our snapshot of the Google database; 15.5% of the cells are non-zero for data-based and only 1.0% for knowledge-based.

[1] **https://radimrehurek.com/gensim/**

[2] **https://code.google.com/p/word2vec/**

| | REQSEQ | | MULTLAB | |
|---|---|---|---|---|
| | Personal | Generic | Personal | Generic |
| Cluster (baseline) | 42.8 | 10.5 | 55.1 | 24.0 |
| +QryEnr | 44.0 | 11.0 | 56.1 | 27.4 |
| +AppSim | 42.8 | 14.8 | 55.1 | 29.2 |
| +QryEnr+AppSim | **44.0** | **15.4** | **56.1** | **38.2** |
| Neighbor (baseline) | 50.8 | 23.8 | 51.3 | 19.1 |
| +QryEnr | 54.9 | 26.2 | 57.0 | 22.9 |
| +AppSim | 50.8 | 30.7 | 51.3 | 24.7 |
| +QryEnr+AppSim | **54.9** | **32.7** | **57.0** | **30.3** |

**Table 1. Weighted average $F_1$ score (%) on test set across 14 participants, using bag-of-word features. Average number of clusters, $K_C$, in the cluster-based approach is $7.0 \pm 1.0$ for generic models, and $7.1 \pm 1.6$ for personalized models. The reported numbers are average performance of 20 K-means clustering results. $K_N$ in the neighbor-based condition is $18.5 \pm 0.4$ for generic models and $4.9 \pm 1.4$ for personalized models. AppSim is rule-based.**

#### *Results*

We compare the apps suggested by our model with the ones actually launched by users (Table 1). This prediction task is difficult; in our corpus, on average each user has 19 unique apps and 25 different sequences of apps. The upper part of the Table corresponds to the cluster-based intention model, the lower part to the neighbor-based intention model. Within each approach, intention realization strategies (QryEnr, AppSim) and their combination are shown.

Bringing more post-initiate information (i.e. set composition) into the clustering process improves performance [27]. But we did not observe better performance for the cluster-based model relative to the neighbor-based model. When the REPSEQ realization strategy is adopted, neighbor-based intention yields a better $F_1$ score. It is possible that REPSEQ is sensitive to the selection of similar interactions. Arguably, an input may fall close to the intention boundary in the cluster-based setting, which indeed is closer to some interactions on the other side of the boundary as opposed to the ones within the same intention cluster. On the other hand, the MULTLAB approach shows relatively consistent performance in both cluster- and neighbor-based settings, indicating ro-

| | REQSEQ | | | MULTLAB | | |
|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| Baseline | 33.3 | 18.9 | 23.8 | 45.8 | 12.3 | 19.1 |
| Rule | 43.3 | 24.3 | 30.7 | 59.4 | 15.9 | 24.7 |
| Knowledge | 41.8 | 22.3 | 28.7 | 53.0 | 14.6 | 22.6 |
| Data | 38.1 | 21.2 | 27.0 | 54.6 | 13.9 | 21.7 |
| Combine | **44.7** | **25.0** | **31.7** | **61.0** | **16.4** | **25.5** |

**Table 2. Comparison of different AppSim approaches on neighbor-based intention in a generic model. Precision, recall and $F_1$ score are reported. For the data-driven method, the vector dimension $D = 500$.**

bustness and self-adaptability with respect to the choice of interactions for similar intentions.

In Table 1, the fact that QryEnr improves $F_1$ in all conditions indicates that semantic similarity among words can effectively address the *language-mismatch* problem. In addition, although AppSim has no effect on personalized models, it addresses the *app-mismatch* issue in generic model in an intuitive way. Combining QryEnr and AppSim methods together (denoted as "+QryEnr+AppSim") consistently achieves the best performance on $F_1$. Further inspection shows that QryEnr improves recall while AppSim improves both precision and recall. As we expected, the generic intention model is inferior to the personalized model for our data. We anticipate that with a larger user community (and perhaps grouping of similar users instead of a single generic model) we will observe better performance. Note that, IAs will face a cold-start problem before sufficient data has been accumulated for specific users. Thus, generic models will always be required to ensure reasonable communication.

Table 2 compares the differences across AppSim methods. Applying AppSim improves the baseline in the generic model. The rule-based approach outperforms other two methods, although it requires filters. This is probably due to the sparseness of similarity matrices in the knowledge- and data-based approaches. Nevertheless, combining three similarity scores yields the best performance, showing the effectiveness of leveraging inter-app similarity for this task.

### Study 2: Intention Representation in Natural Language

We used Rapid Automatic Keyword Extraction (RAKE[3]) algorithm [1], an unsupervised, language- and domain-independent extraction method, reported to outperform other unsupervised method such as TextRank [19, 11] in both precision and $F$ score. In RAKE, we required that 1) each word have 3 or more characters; 2) each phrase have at most 3 words; and that 3) each key word appear in the text at least once. We did not tune these parameters. We used 3 individual resources and 2 combinations, reflecting constraints on the availability of different contexts in real-life. The three individual resources are manual transcription of user utterances from their dialogs (MANUAL), ASR transcriptions (ASR)

[3]`https://www.airpair.com/nlp/keyword-extraction-tutorial`

| MANUAL | ASR | DESC | DESC + ASR | DESC + MANUAL |
|---|---|---|---|---|
| 20.0 | 20.3 | 11.3 | 29.6 | 29.1 |

**Table 3. Mean number of phrases generated using different resources**

thereof and high-level task descriptions (DESC). The number of key phrases that could be generated by each resource or their combination depends on resource size (Table 3).

We asked 6 users to first review and refine their own clusters, by showing them all cluster members. Toaid rercall we displayed, 1) context e.g., location, time; 2) task descriptions (e.g., "planning a dinner"), 3) dialogs produced and 4) apps involved. Users could decide whether to split each cluster into subgroups. Then, based on the refined clusters, we generate ranked lists of key phrases using the different resources. Users were asked to provide binary judgment for each phrase in the list (randomized) indicating whether it correctly summarized all the activities in the current (refined) cluster.

To focus on a practical goal, we used Mean Reciprocal Rank (MRR)—"how deep the user has to go down a ranked list to find one descriptive phrase?" Average MRR was 0.64 across different resources and their combinations, meaning that on average the user can find an acceptable phrase in the top 2 items shown; although MRR is lower when individual resource was used, an ANOVA did not show significant differences between resources (and their combinations). Other metrics such as Precision at position $K$ or Mean Average Precision at position $K$ shows DESC+ASR and DESC+MANUAL do best, especially when $K$ is larger. Results indicate that having a task description is useful.

To conclude, if the IA can observe a user's speech commands or elicit descriptions from user (ideally both), it can generate understandable activity references and could communicate more effectively than using alternatives (e.g. lists).

## CONCLUSION AND FUTURE WORK

We present a framework, HELPR, that implicitly learns from past interactions to map high-level expressions of goals (e.g., "go out with friends") to specific functionality (apps) available on a smart device. The proposed agent uses language produced by user to identify interactions similar to the current input. A set of domains/apps can be proposed from past experience and used to support current activities. This framework is also capable of generating natural language references to a past experience cluster. As a result, the communication channel would have greater transparency, supporting timely recovery from possible misunderstandings.

Our long-term goal is to create agents that observe recurring human activities, figure out the underlying intentions and then provide active support through language-based interaction (in addition to allowing the user to explicitly teach the agent about complex tasks). The value of such an agent is that it can learn to manage activities on a level more abstract than provided by app-specific interfaces and would allow users to build their own (virtual) applications that combine the functionality of existing apps.

## REFERENCES


1. Berry, M. W., and Kogan., J. Text mining: applications and theory (2010).
2. Bohus, D., and Rudnicky, A. I. Constructing accurate beliefs in spoken dialog systems. In *Proceedings of IEEE Workshop on Automatic Speech Recognition and Understanding* (2005).
3. Chen, Y.-N., and Rudnicky, A. I. Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings. In *Proceedings of 2014 IEEE Spoken Language Technology Workshop (SLT)*, IEEE (2014), 590–595.
4. Chen, Y.-N., Sun, M., and Rudnicky, A. I. Leveraging behavioral patterns of mobile applications for personalized spoken language understanding. In *Proceedings of 2015 International Conference on Multimodal Interaction (ICMI)* (2015).
5. Chen, Y.-N., Sun, M., and Rudnicky, A. I. Matrix factorization with domain knowledge and behavioral patterns for intent modeling. In *NIPS Workshop on Machine Learning for SLU and Interaction* (2015).
6. Chen, Y.-N., Sun, M., Rudnicky, A. I., and Gershman, A. Unsupervised user intent modeling by feature-enriched matrix factorization. In *Proceedings of The 41th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)* (2016).
7. Duda, R., Hart, P., and Stork, D. *Pattern Classification*. John Wiley and Sons, 2012.
8. Fiscus, J. G. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (rover). In *Proceedings of Automatic Speech Recognition and Understanding Workshop (ASRU)* (1997), 347–352.
9. Ganesan, K., Zhai, C., and Han, J. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd international conference on computational linguistics (COLING)*, ACL (2010), 340–348.
10. Harrison, C., Xiao, R., Schwarz, J., and Hudson, S. E. Touchtools: leveraging familiarity and skill with physical tools to augment touch interaction. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2014), 2913–2916.
11. Hulth, A. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of the 2003 conference on Empirical methods in natural language processing (EMNLP)*, ACL (2003), 216–223.
12. Le, Q. V., and Mikolov, T. Distributed representations of sentences and documents. In *ICML* (2014).
13. Li, Q., Tur, G., Hakkani-Tur, D., Li, X., Paek, T., Gunawardana, A., and Quirk, C. Distributed open-domain conversational understanding framework with domain independent extractors. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, IEEE (2014), 566–571.
14. Lin, B.-s., Wang, H.-m., and Lee, L.-s. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proceedings of 1999 IEEE Workshop on Automatic Speech Recognition and Understanding Workshop (ASRU)*, vol. 99 (1999), 4.
15. Liu, F., Flanigan, J., Thomson, S., Sadeh, N., and Smith, N. A. Toward abstractive summarization using semantic representations. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)* (2015).
16. Lucchese, C., Orlando, S., Perego, R., Silvestri, F., and Tolomei., G. Identifying task-based sessions in search engine query logs. In *Proceedings of the fourth ACM international conference on Web search and data mining*, ACM (2011), 277–286.
17. Lunati, J.-M., and Rudnicky, A. I. Spoken language interfaces: The OM system. *CHI91 Human Factors on Computing Systems* (1991).
18. Medelyan, O. Human-competitive automatic topic indexing. In *Thesis Dissertation, University of Waikato* (2009).
19. Mihalcea, R., and Tarau, P. Textrank: Bringing order into texts. In *ACL* (2004).
20. Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In *Proceedings of Workshop at International Conference on Learning Representations (ICLR)* (2013).
21. Nakano, M., Sato, S., Komatani, K., Matsuyama, K., Funakoshi, K., and Okuno, H. G. A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In *SIGdial Workshop on Discourse and Dialogue (SIGDIAL)*, Association for Computational Linguistics (2011), 18–29.
22. Rudnicky, A. I., Lunati, J.-M., and Franz, A. M. Spoken language recognition in an office management domain. In *Proceedings of International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, IEEE (1991), 829–832.
23. Ryu, S., Song, J., Koo, S., Kwon, S., and Lee, G. G. Detecting multiple domains from users utterance in spoken dialog system. In *Proceedings of the International Workshop on Spoken Dialogue Systems (IWSDS)* (2015).
24. Shen, X., Tan, B., and Zhai., C. Implicit user modeling for personalized search. In *Proceedings of the 14th ACM international conference on Information and knowledge management*, ACM (2005), 824–831.
25. Sun, M., Chen, Y.-N., and Rudnicky., A. I. Learning OOV through semantic relatedness in spoken dialog systems. In *$16^{th}$ Annual Conference of the International Speech Communication Association (Interspeech)* (2015).

26. Sun, M., Chen, Y.-N., and Rudnicky, A. I. Understanding user's cross-domain intentions in spoken dialog systems. In *NIPS Workshop on Machine Learning for SLU and Interaction* (2015).

27. Sun, M., Chen, Y.-N., and Rudnicky, A. I. HELPR: A framework to break the barrier across domains in spoken dialog systems. In *International Workshop on Spoken Dialog Systems* (2016).

28. Tibshirani, R., Walther, G., and Hastie., T. Estimating the number of clusters in a data set via the gap statistic. In *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* (2001), 411–423.

29. Witten, I. H., Paynter, G. W., Frank, E., Gutwin, C., and Nevill-Manning, C. G. Kea: Practical automatic keyphrase extraction. In *Proceedings of the fourth ACM conference on Digital libraries* (1999), 254–255.

30. Zhao, R., Papangelis, A., and Cassell, J. Towards a dyadic computational model of rapport management for human-virtual agent interaction. In *Intelligent Virtual Agents*. 2014, 514–527.