Learning User Intentions Spanning Multiple Domains

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 applications in smart environments and get assistance with specific tasks. Current intelligent agents (IAs) tend to be limited to specific applications. In order to engage in more complex activities users have to directly manage a task that may span multiple applications. An ideal personal IA would be able to learn, over time, about these tasks that span different resources. This paper addresses the problem of multi-domain task assistance in the context of spoken dialog systems. We propose approaches to discover users' high-level intentions and using this information to assist users in their task. We collected real-life smart phone 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); Intelligent assistant (IA); 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

Environments, such as a home, can host smart objects/devices where each typically operates in a specific domain (for example, climate control or security). Each such object, by design manages few domains, usually one. For example, a fridge may support a grocery domain by tracking vegetables inside of it, perhaps additionally helping to compose a shopping list. It might even be configured to support sharing of information to other domains known to it, but such functionality would not be scalable and might lack potentially desirable adaptive features.

In contrast, users often mentally arrange tasks that span domains and easily manage the information shared among them. However, even if we assume that environment information is aggregated into a handheld device (e.g., phone) in the form of apps, the process of launching apps one by one may be time-consuming and difficult for users, especially for elders and ones 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 intelligent agents (IAs) to automatically help us organize tasks across domains (or, apps) given a user's request expressed, in language, at the level of intentions. For example, upon receiving "can you help me plan a grocery shopping trip?", the IA should determine what foods are out of stock (FRIDGE), the next bus to a nearby supermarket (NAVIGATION) and finally the locations of the food inside the supermarket (AISLEFINDER).

Conventional dialog systems operate in specific domains such as restaurant [31, 11], bus information [21] or event arrangement [19]. Multi-domain dialog systems have been studied in the past [16, 22, 15, 18, 14, 4, 24, 5, 6, 7], but they typically lack the capability of understanding the user's goal or high-level intention. As a result such a system has certain drawbacks (see example in Fig 1): 1) it passively selects *one* domain at a time given the user request; 2) it has no expectation of the next domain expected by the user at the task level; 3) it does not maintain a shared context across domain boundaries. Consequently, the multi-domain conversation will not be as natural/efficient as a human assistant would provide. The IA should assist human users in interacting with multiple domains, as in the following use cases: 1) during conversation, our model can use the current context to predict user's next action as well as his high-level intention [27]; 2) the IA should understand the user's high-level intention (e.g. as stated at the start of the conversation) and coordinate existing domains to accomplish this intention [28, 29].

In this paper, we summarize our findings from previous work, followed by the remaining challenges in this multi-domain agent setup.

DATA COLLECTION

To let the agent learn how human users coordinate existing functionalities for complex tasks, there are two ways: 1) users can explicitly instruct/author such procedural knowledge [23, 1]; 2) the agent can observe how users perform such tasks and learn. The first approach may rely on the agent's capability to comprehend sequential instructions, as well as user' capability to thoroughly define a task via language. In this work, we take the second approach. We use a smartphone to allow us to investigate cross-domain task management without having to deal with the complexity of a fully situated implementation.

IUI 2016 Workshop: Interacting with Smart Objects, March 10th, 2016, Sonoma, CA, USA

S: What can I do for you?

U: Could you arrange a dinner for me and my friends?

S: Sorry I don't understand that. What can I do for you?

U: Can I book a table for three in Tākō downtown for this Friday?

...

S: OK. What can I do for you next?

U: Show me the bus from here.

S: Where is your destination please?

U: Tākō downtown Pittsburgh.

...

S: What should I do next?

U: Send the bus route to Carrie and Peter.

S: OK. Sending the following message to Carrie and Peter: "the bus route".

A: What can I do for you?

U: Could you arrange a dinner for me and my friends?

A : What kind of food do you prefer?

U: Mexican?

A : How about Tākō? I can book a table for you.

U: Sounds good! Can I take a bus there?

A : 61 A/B/C/D can take you there. Do you want to send this to your friends?

U: Great! Send it to Carrie and Peter.

A: OK. The bus route 61 has been sent.

Figure 1. Left: example dialog between user (U) and a classic multi-domain dialog system (S); Right: example dialog between user (U) and human assistant (A).

We logged real-life interactions at app-level (app invocation + when + where), segmenting a day's log into episodes (as separated by periods of 3 minute inactivity). Each episode could contain more than one app invocation. We asked users to annotate each episode to: 1) group apps used for a particular goal; and 2) describe the goal in language. Meta information such as day, time, location was shown to the user to aid recall. Users were asked to re-enact the smart phone interaction by speaking with a Wizard-of-Oz dialog system. The participants were not required to follow the order of the applications they used on the smart phones. Other than for remaining on-task, we did not constrain expression. The wizard (21-year-old male native English speaker) was instructed to respond directly to a participant's goal-directed requests and to not accept out-of-domain inputs. An example of annotation and Wizard-of-Oz dialog is shown in Figure 2.

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. 12 were native English speakers. On average, each user interacts with 19 different apps. Across 14 users, a total of 132 apps were used. Details of the collection are provided in [25].

DOMAIN TRANSITION

As mentioned earlier, users can mentally coordinate a set of domains to accomplish complex tasks. However, this would require user to manually launch the next domain through speech, touch-screen or other modalities. Ideally, we would like the agent to have some expectation of the follow-up domain such that the transition can be smooth and easy for the user. In our previous work, we built context-based model to predict the next domain a user would interact with [27]. The agent could use this expectation to warm up the predicted app in the background, forward current information about the interaction so far to this app, or even proactively fetch the information from the next app and offer to the user.

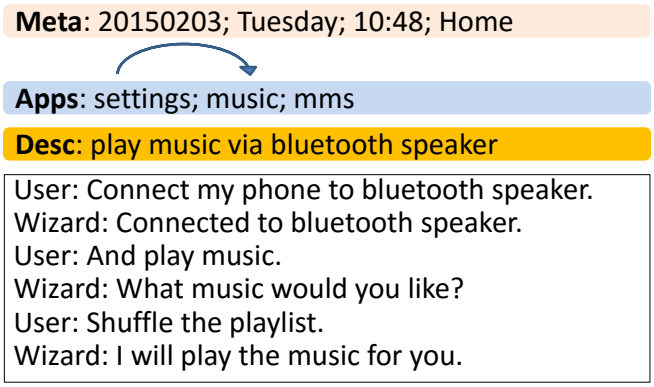

Figure 2. User annotation: 1) user connected SETTINGS and MUSIC; and 2) user noted that these two apps were used to *play music via bluetooth speaker*. Wizard-of-Oz dialog was collected and manually transcribed.

Rank	App Prediction	Intention Prediction
1	*Lang+App*	*All*
2	*All*	*Meta+App*
3	*Meta+App*	*Lang+App*
4	Lang	App
5	App	Lang
6	Majority	Location
7	Time	Majority
8	Day	Meta
9	Location	Time
10	Meta	Day

Table 1. Ranked features for prediction tasks without goal statement. Meta is a combination of time, day and location.

The contexts for prediction include 1) time (hours based on 24 hour clock), day (Monday, Tuesday, ..., Sunday), location (street names e.g, "Forbes Ave"); 2) previously launched app; 3) (noun and verb) words in user utterance. The intuition behind these contexts are: 1) people would have different tasks in different places or time; 2) what information people obtained (in certain app via certain speech command) would indicate what he might want to seek next. From our experiment, we found that previous app and user utterance are very informative by using multi-class (i.e., app ids) classification models. The rank of the result is shown in the left side of Table 1.

INTENTION PREDICTION

A user might not explicitly express their ultimate goal/intention (e.g., "plan a dinner with friends"). The agent might need to infer the user's ultimate intention to provide timely assistance. Knowing that the user's intention is to "schedule a meeting", it could reference how this user (or others) accomplishes this task. Note that sharing this information can improve the communication channel transparency: The agent can say "I think you want to *plan a dinner*. Let's find a restaurant for you first." In this way, the agent can reveal its (mis-)understanding of the task at hand, allowing the user to redirect it.

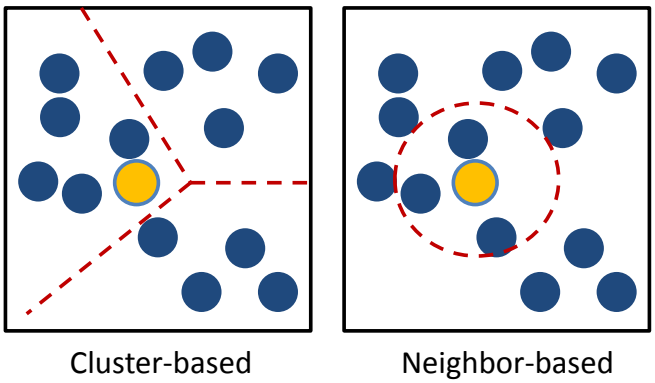

Figure 3. Cluster-based vs. Neighbor-based intention definition.

There are two cases in predicting a user's high-level intention. First, when the user does not explicitly state his intention, the agent can infer it by context (time, day, location, apps and user utterances). On the other hand, the user may directly express the intention via language at the beginning of the interaction. The agent could map that expression to an intention. Regardless of the different use cases, this process of inferring user's intention has two phases: 1) define a finite set of intentions and 2) recognize the intention given the input (either the context or the user's utterance at the outset).

Intention Definition

We let our agent automatically cluster seen interactions into K_C groups, each representing one intention. Bag-of-words features extracted from each interaction are words (lemmatized nouns and verbs) in user-generated language (i.e., task description and user utterances) as well as other contexts. The number of clusters K_C can be automatically optimized by using gap statistic [30]. We call this cluster-based intention definition.

We conducted a user study to examine the effectiveness of this cluster-based approach [28]. Users were shown the cluster members, each with the produced dialog, apps involved, task description provided by user. We asked each user to rate their agreement with the statement that the tasks shown in each cluster are essentially of the same nature. On average, we obtained 4.2 out of 5.0, where 5.0 indicates strong agreement. An example of clustered user tasks are shown in Table 2.

Similarly, we can use the K_N nearest neighbors of the input to denote the current intention (neighbor-based definition). The difference is illustrated in Fig 3. In this approach, K_N was set to the square root of the number of training examples [8]. The advantage of cluster-based approach is that the agent has awareness of the typical tasks a user performs everyday. This can be useful in the future when the agent list a few possible intentions for grounding, to prevent potential misunderstandings.

Intention Recognition

In this section, we introduce two use cases of intention recognition: one with ultimate goal/intention expressed and one without. When the user does not express his goal, the agent needs to recognize the goal implicitly from other contexts. On the other hand, if the goal is directly expressed (usually at the beginning of the interaction), the agent needs to understand it. In the following, we briefly note our findings when no goal statement is provided. We then focus more closely on understanding intention from explicitly expressed goal statements.

Use Case 1: without goal statement

Multi-class classification technique is adopted to recognize the user intention. The input are current contexts — 1) time, location; 2) previously launched app; 3) user utterance. Similar to the results in app prediction, last app and user utterance outperform other contexts when predict user intention. Fusing all contexts together yields the best performance. Ranks of individual feature and combinations of features are shown in the right side of Table 1.

Use Case 2: with goal statement

The second use case where user initiates the conversation by a high-level command (e.g., "please organize a meeting for me") is illustrated in Fig 4. Let's take cluster-based intention definition as an example. We first segment the semantic space constructed from past interactions based on user's high-level commands as well as other contexts (red dashed lines in Fig 4). During execution time, user utters a new command (yellow) which would be mapped to a certain cluster within this semantic space. Thus, we can find past experience of similar nature to the input. By using the information provided by such experience, i.e., how user previously performed these task with domains, our agent is able to effectively map the new command to a set of supportive domains in the following ways:

- **Representative Sequence** (REPSEQ): We can combine the individual app sequences of the set members into a single app sequence that represents a common way of surfacing the intention. An example is shown in Fig 5. We used ROVER to implement this majority vote [9].
- **Multi-label Classification** (MULTLAB): We can treat this problem as associating multiple labels (app ids) to the input command, given the training instances of cluster members (or neighbors). We used SVM with linear kernel.

In this use case, we have the following obstacles: 1) people use different language to describe tasks of similar nature (e.g., "take a picture" vs. "snap a photo"); 2) people use different domains/apps for essentially the same functionality (e.g., GMAIL vs. MESSENGER for contacting someone). The first obstacle holds even for the same user. We adopted the following techniques to solve these problems, with the goal of improving the system's prediction performance especially for a model trained from a generic group of users:

- **Query Enrichment** (QryEnr): We expand the query/ command by incorporating words related to it semantically. QryEnr can reduce the likelihood of seeing sparse input feature vector due to out-of-vocabulary [26] words. The algorithm is shown in Algorithm 1. In short, each word w_i in the lemmatized query Q yields mass increases for N semantically close words in the feature vector f.

Cluster	Item Examples (task descriptions supplied by participant)
1	"Picture messaging XXX", "Take picture and send to XXX"
2	"Look up math problems", "Doing physics homework", "Listening to and trying to buy a new song"
3	"Talking with XXX about the step challenge", "Looking at my step count and then talking to XXX about the step challenge"
4	"Playing [game] spiderman", "Allocating memory for spiderman"
5	"Using calculus software", "Purchasing Wolfram Alpha on the play store"
6	"Texting and calling XXX", "Ask XXX if she can talk then call her"
7	"Talking and sharing with group mates", "Emailing and texting group members"

Table 2. Intention clustering of tasks based on utterances, with typical descriptions.

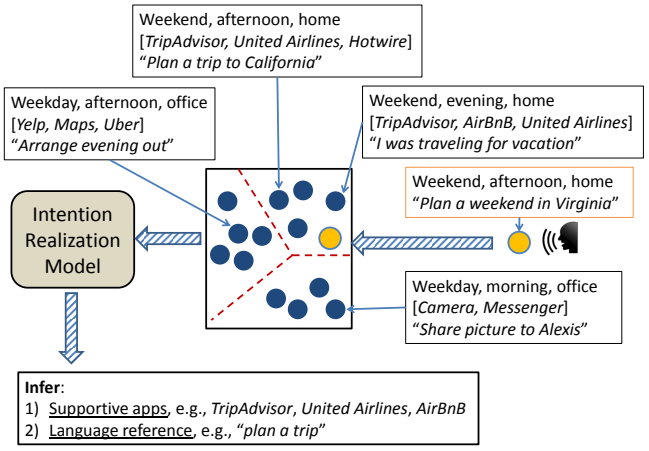

Figure 4. Intention understanding (cluster-based) and realization example. Solid nodes denote past interactions (blue) and current input (yellow).

- **App Similarity** (AppSim): Similarity between two apps is measured in the following ways:
  - Data-driven: App descriptions from the Google Play Store can be projected into a high-dimensional semantic space to compute similarity. We used doc2vec [13] via gensim[1] trained on 1 million app descriptions. Cosine similarity can then be computed given any two apps. Most objects will have associated descriptive materials and we expect this approach can scale accordingly.
  - Knowledge-driven: The Google Play store provides a finite ranked list of "similar apps" for each entry. We used reversed rank $(1/r)$ as similarity.
  - 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". Here, we applied a manually constructed list of 50 filters (e.g., "com", "android", "lge") on package names. Then we compute Edit Distance based similarity.

[1] `https://radimrehurek.com/gensim/models/doc2vec.html`

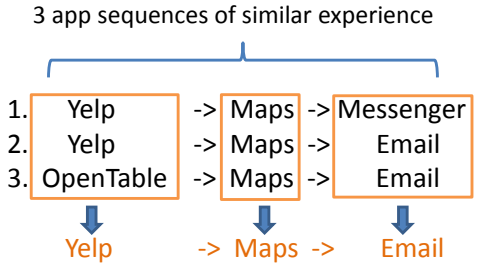

Figure 5. Example of Representative Sequence approach

Algorithm 1 Query Enrichment

Require: lemmatized words of the query $Q = \{w_1, ..., w_{|Q|}\}$ and their counts $C = \{c_1, ..., c_{|Q|}\}$; training vocabulary V; bag-of-word feature vector $Q_f = \{f_1, ..., f_{|V|}\}$ constructed on Q; the word semantic relatedness matrix M; the number of semantically similar words N to be extracted for each word in Q;

Ensure: an enriched bag-of-word feature vector

1: **for** each $w_i \in Q$ **do**
2: Use M to find the N closest words $V_N = \{v_1, ..., v_N\} \in V$;
3: **for** each $v_j \in V_N$ **do**
4: $f_j = f_j + M_{i,j} \times c_i$
5: **end for**
6: **end for**
7: **return** f;

We compare the system-generated apps with the ones users actually launched to compute precision, recall and F_1 score. The results are shown in Table 3. Our main finding is that the original gap between personalized model and generic model can be effectively reduced by adopting QryEnr and AppSim techniques, while the personalized model performance (i.e. the upper bound of our agent) can be improved as well. This

	RepSeq		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	—	14.8	—	29.2
+QryEnr+AppSim	—	**15.4**	—	**38.2**
Neighbor (baseline)	50.8	23.8	51.3	19.1
+QryEnr	**54.9**	26.2	**57.0**	22.9
+AppSim	—	30.7	—	24.7
+QryEnr+AppSim	—	**32.7**	—	**30.3**

Table 3. 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 ± 1.0 for generic models, and 7.1 ± 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 ± 0.4 for generic models and 4.9 ± 1.4 for personalized models. AppSim is rule-based.

	RepSeq			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 4. 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 $D = 500$.

result shows that, after deployment, if the agent keeps observing the user performing tasks, it can learn to assist the user in the future. Even if this is a new user, or out of privacy concern, given insufficient user data, the agent can use generic model obtained from other users.

We varied ways to compute app similarity. The result is shown in Table 4. As we can see, rule-based approach outperforms the other approaches. It is not clear whether this is due to the coverage issue in the other two methods: since vendor apps do not have entries in our snapshot of Google Store database, 15.5% of the cells of the data-driven similarity matrix are non-zero. For knowledge-driven matrix, only 1.0% are non-zero. Combining three similarity measurements together provides the best performance.

LANGUAGE REFERENCE

To reveal the agent's understanding of user's intention can improve the channel transparency. One useful modality is for the agent to verbally convey such understanding (e.g., "I think you want to *plan a trip*"). We adopted keyphrase extraction [2] on user-generated language to generate a ranked list of phrases. We used Rapid Automatic Keyword Extraction (RAKE[2]) algorithm [2], an unsupervised, language- and domain-independent extraction method, reported to outperform other unsupervised method such as TextRank [17, 12] 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) thereof and high-level task descriptions (Desc). The number of phrases generated from different language resources (and their combinations) are shown in Table 5.

[2] `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 5. Mean number of phrases generated using different resources. Manual: manual transcription of user utterances. ASR: Google speech recognition transcription of user utterances. Desc: user description of the task.

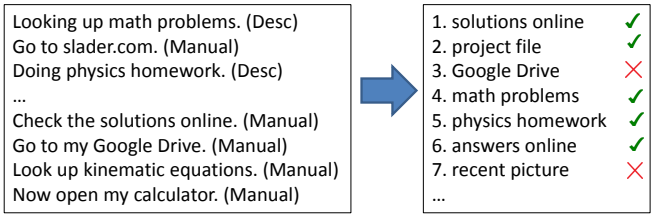

Figure 6. Key phrases (ranked) extracted from user-generated language, with user judgment.

We selected 6 users to first review their own clusters, by showing them all cluster members with 1) apps used in the member interaction; 2) dialog reproduced; 3) meta-data such as time, day, address, etc. We let them judge whether each individual phrase (the order is randomized) generated by the system summarized all the activities in the cluster (binary judgement). See example in Fig 6.

We found that, on average, users would find an acceptable phrase within top 2 of the list (average Mean Reciprocal Rank $= 0.64$). This demonstrates that, the agent can generate understandable activity references. An ANOVA did not show significant differences between resources. With more sensitive metrics MAP@K[3] (Mean Average Precision at position K) and P@K (Precision at position K) metrics, Desc+ASR and Desc+Manual do best. The improvement becomes significant as K increases: having a user-generated task description is very useful.

REMAINING CHALLENGES

We have shown that it's possible to build models to infer a user's intention and use this information to activate the set of domains that will allow the user to accomplish the high-level goal. At the same time, we have found that an agent can use the user's high-level descriptions to generate language referencing this goal in conversation. Nevertheless there are still

[3] MAP@K $= \sum_{k=1}^{K} precision(k) * relevance(k)/K$

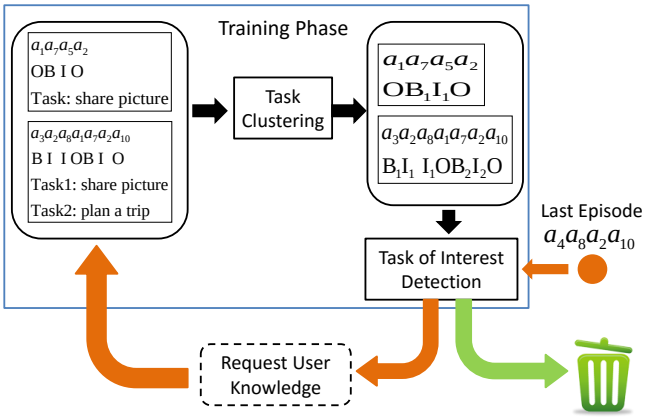

Figure 7. Pipeline for growing complex task inventory

remaining challenges to solve before the agent assistant is able to actively help the user in their high-level activity.

One challenge is having to ask the user to explain what they are doing is intrusive and perhaps not realistic. This would be time-consuming and intrusive in realistic settings. Ideally, the agent should have the ability to balance the cost of an interruption and the expected value of the information to be gained and act accordingly. In practical terms, the agent could observe the user for some period of time to first identify activities that appear to recur. Only when such activities are recognized should the agent ask for a description. We expect that the agent would continue to monitor activities to detect changes or even propose ones itself.

Another challenge is the need for the agent to manage activity-level context so that relevant information can be transferred between apps; for example, passing the address of a restaurant from a reviews app to an app that will provide directions on how to get there. Unless perhaps the apps were developed by the same vendor, its unlikely that similar concepts will be easy to match across different domains. But doing so is necessary for maintaining a context.

Data Gathering

There are explicit and implicit ways to acquire knowledge about a user's high-level intentions so that the inventory of intentions can evolve over time. First, user could explicitly teach the agent about an activity by saying "To ***plan a trip***, you should find a cheap flight from PRICELINE and then look for 3-star hotel in downtown by using HOTWIRE ..." Such instructable agents has been studied in dialog setup [23, 1]. The difficulties lie in the dependence on the agent's capability to comprehend complex language instructions. Alternately, the user could say "***Agent, watch this***" allowing the agent to observe an activity and then ask the user for more information as needed. This initiative command ("*Agent, watch this*") can be further omitted if the agent is capable of segmenting stream of events into meaningful sessions, e.g., thresholding the idle time.

A proposed pipeline for this process of knowledge acquisition is shown in Fig 7. We assume that the initial inventory has full annotation from user for each task — 1) apps (e.g., $a_1a_7a_5a_2$ where the indicies indicate the app id); 2) IOB labels [20] assigned to the apps to distinguish noise (O's) from content (B's and I's); 3) the user's description of what the task is about.

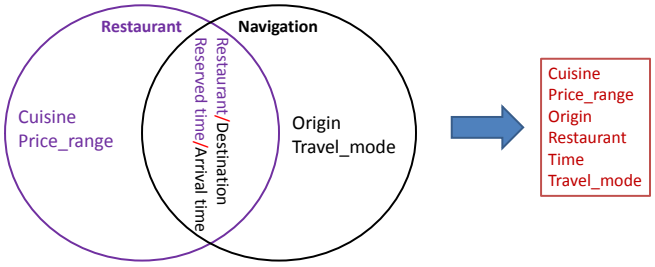

Figure 8. Illustration of overlapping domain knowledge. The size of shared context (red) is less than the sum of concepts in individual domain (purple and black).

After deployment, the agent first clusters seen interactions (complex tasks in the inventory) into groups and indexes the IOB labels in each interaction accordingly. For example, tasks similar to "sharing picture" are grouped into cluster 1. Then their corresponding IOB labels would be indexed (e.g., $OBIO \rightarrow OB_1I_1O$). Next, the agent detects a task of interests (ToI) — either a recurrence of seen task type or an anomaly which does not conform to the automatically learned tasks (i.e., normal behavior) [3]. Either way, the system can take appropriate actions to add the current task back into the task inventory during the execution time (path in orange) is elaborated in Table 6.

Sharing Context

The necessary communication skills to request necessary information concepts/slots are already provided by domain experts (e.g., "`request_destination`" in NAVIGATION domain, "`inform_restaurant_review`", "`request_price_range`" in RESTAURANT domain). It is up to the agent to mix these skills into one conversation. However, the concepts required by different domains may overlap. For example, `destination` in NAVIGATION domain may be implied by the `restaurant` in RESTAURANT domain. More intelligently, the `arrival_time` in NAVIGATION can be computed as `reserved_time` - 10min by understanding that user tends to arrive 10min earlier. Therefore, it is important for the agent to have a shared context in order to avoid requesting information it has already possessed.

An illustration is shown in Fig 8. Two domains (RESTAURANT and NAVIGATION) have a few concepts in common but with different names. When collectively serving a common user intention ("plan a dinner"), knowing the value of `Restaurant` slot induces the value of `Destination`. Thus, this problem can be formalized as follows: Given the union of concepts in D different domains $C = C_1 \cup C_2 \cup ... \cup C_D$ and the concepts already filled in $C_F \in C$, for the target concept c_t find the source concept $c_s = \arg\max_{c_i \in C_F} R(c_i, c_t|\text{Intention})$, where function R mea-

ToI Type	Confidence	System Action	Example Sub-dialog
Recurrence	High	Add to seen interactions	N/A
Recurrence	Mild	Confirm with user	"I think you were *planning a party*, am I right?"
Recurrence	Low	Request user annotation	"Could you tell me what you just did, is it one of `[list of tasks]`?"
Anomaly	N/A	Request user annotation	"I think you were doing something new, could you teach me?"

Table 6. System actions based on classification and confidence

sures the relatedness (semantic similarity) between two concepts given the ultimate user intention. A further filtering function has to be applied to either use the source concept c_s (e.g., `restaurant`) as the target concept (`destination`) or discard it. A perfect R measurement plus the filtering function would resolve the inter-domain as well as intra-domain redundancy.

The following approaches can be adopted to learn the semantic relatedness function $R(c_s, c_t|\text{Intention})$ where we assume c_s and c_t are from different domains:

1. Rule-based: In a multi-domain conversation, if the values of c_s and c_t coincide, they are probably of similar nature;
2. Data-driven: We can embed concepts C in the training dialog corpus. Thus, c_s and c_t can be projected to a semantic space and their similarity can be computed.

We believe that this class of information could be pooled across users: identifying the right mappings in principle needs to be done only once for any given pair of apps, with extensions being inferred through transitivity. At this point, this is speculative. But we believe that it can be part of a strategy for establishing an operational ontology across apps.

CONCLUSION

We present a framework that will allow an agent to implicitly learn from past interactions to map high-level expressions of goals (e.g., "go out with friends") to specific functionalities (apps) available in a smart environment. 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 past experience clusters. As a result, the communication channel would have greater transparency, supporting timely recovery from possible misunderstandings. The value of such an agent is that it can learn to manage activities on a level more abstract than provided by object-specific interfaces and would allow users to build their own (virtual) apps that combine the functionalities of existing objects.

ACKNOWLEDGMENTS

This work is partially funded by Yahoo! InMind project and General Motors Advanced Technical Center. We thank Zhenhao Hua for implementing the logger app. We thank Yulian Tamres-Rudnicky and Arnab Dash for collecting the data.

REFERENCES

1. Azaria, A., Krishnamurthy, J., and Mitchell, T. M. Instructable intelligent personal agent. In *AAAI* (2016).
2. Berry, M. W., and Kogan., J. Text mining: applications and theory (2010).
3. Chandola, V., Banerjee, A., and Kumar, V. Anomaly detection: A survey. In *ACM computing surveys (CSUR)* (2009).
4. 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.
5. 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).
6. 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).
7. 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).
8. Duda, R., Hart, P., and Stork, D. *Pattern Classification*. John Wiley and Sons, 2012.
9. 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.
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. Hastie, H., Aufaure, M.-A., Alexopoulos, P., Bouchard, H., Breslin, C., Cuayhuitl, H., Dethlefs, N., Gaic, M., Henderson, J., Lemon, O., Liu, X., Mika, P., Mustapha, N. B., Potter, T., Rieser, V., Thomson, B., Tsiakoulis, P., Vanrompay, Y., Villazon-Terrazas, B., Yazdani, M., Young, S., and Yu, Y. The Parlance mobile application for interactive search in english and mandarin. In *SIGDIAL* (2014).
12. 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.

13. Le, Q. V., and Mikolov, T. Distributed representations of sentences and documents. In *ICML* (2014).

14. 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.

15. 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.

16. Lunati, J.-M., and Rudnicky, A. I. Spoken language interfaces: The OM system. *CHI91 Human Factors on Computing Systems* (1991).

17. Mihalcea, R., and Tarau, P. Textrank: Bringing order into texts. In *ACL* (2004).

18. 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.

19. Pappu, A., Sun, M., Sridharan, S., and Rudnicky, A. I. Situated multiparty interaction between humans and agents. In *Human-Computer Interaction* (2013).

20. Ramshaw, L. A., and Marcus, M. P. Text chunking using transformation-based learning. In *Proceedings of the ACL Workshop on Very Large Corpora* (1995).

21. Raux, A., Langner, B., Black, A. W., and Eskenazi, M. LETS GO: Improving spoken dialog systems for the elderly and non-native. In *Eurospeech* (2003).

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. Rudnicky, A. I., Pappu, A., Li, P., Marge, M., and Frisch, B. Instruction taking in the teamtalk system. In *AAAI Fall Symposium: Dialog with Robots* (2010).

24. 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).

25. Sun, M., Chen, Y.-N., Hua, Z., Tamres-Rudnicky, Y., Dash, A., and Rudnicky, A. I. Appdialogue: Multi-app dialogues for intelligent assistants. In *LREC* (2016).

26. Sun, M., Chen, Y.-N., and Rudnicky., A. I. Learning OOV through semantic relatedness in spoken dialog systems. In *16th Annual Conference of the International Speech Communication Association (Interspeech)* (2015).

27. 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).

28. 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).

29. Sun, M., Chen, Y.-N., and Rudnicky, A. I. An intelligent assistant for high-level task understanding. In *IUI* (2016).

30. 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.

31. Young, S. Using POMDPs for dialog management. In *SLT* (2006).