An Intelligent Assistant for High-Level Task Understanding

Ming Sun  Yun-Nung Chen  Alexander I. Rudnicki
School of Computer Science, Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213, USA
{mings, yychen, 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 potential 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/functionalities. 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 particular 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
User: Connect my phone to bluetooth speaker.
Wizard: Connected to bluetooth speaker.
User: And play music.
Wizard: I will play the music for you.

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.

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 model (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 (M vs. “taking”); 2) app choice, e.g., people may use differences such as wrong word choice or morphology (“take” vs. “shoot photo”), sometimes caused by minor differences such as wrong word choice or morphology (“take” vs. “taking”); 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.

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 on the model pre-trained on GoogleNews [20]. Each word $w_i$ 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.ique.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.

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-

<table>
<thead>
<tr>
<th></th>
<th>REQSEQ</th>
<th>MULTLAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Personal</td>
<td>Generic</td>
</tr>
<tr>
<td>Cluster (baseline)</td>
<td>42.8</td>
<td>10.5</td>
</tr>
<tr>
<td>+QryEnr</td>
<td>44.0</td>
<td>11.0</td>
</tr>
<tr>
<td>+AppSim</td>
<td>42.8</td>
<td>14.8</td>
</tr>
<tr>
<td>+QryEnr+AppSim</td>
<td>44.0</td>
<td>15.4</td>
</tr>
<tr>
<td>Neighbor (baseline)</td>
<td>50.8</td>
<td>23.8</td>
</tr>
<tr>
<td>+QryEnr</td>
<td>54.9</td>
<td>26.2</td>
</tr>
<tr>
<td>+AppSim</td>
<td>50.8</td>
<td>30.7</td>
</tr>
<tr>
<td>+QryEnr+AppSim</td>
<td>54.9</td>
<td>32.7</td>
</tr>
</tbody>
</table>

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.

\[1\] https://radimrehurek.com/gensim/
\[2\] https://code.google.com/p/word2vec/
Table 1. 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 $D = 500$.

<table>
<thead>
<tr>
<th></th>
<th>REQSEQ</th>
<th>MULTILAB</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Prec.</td>
<td>Rec.</td>
</tr>
<tr>
<td>Baseline</td>
<td>33.3</td>
<td>18.9</td>
</tr>
<tr>
<td>Rule</td>
<td>43.3</td>
<td>24.3</td>
</tr>
<tr>
<td>Knowledge</td>
<td>41.8</td>
<td>22.3</td>
</tr>
<tr>
<td>Data</td>
<td>38.1</td>
<td>21.2</td>
</tr>
<tr>
<td>Combine</td>
<td>44.7</td>
<td>25.0</td>
</tr>
</tbody>
</table>

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 $D = 500$.

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

<table>
<thead>
<tr>
<th></th>
<th>MANUAL</th>
<th>ASR</th>
<th>DESC</th>
<th>DESC + ASR</th>
<th>DESC + MANUAL</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>20.0</td>
<td>20.3</td>
<td>11.3</td>
<td>29.6</td>
<td>29.1</td>
</tr>
</tbody>
</table>

We asked 6 users to first review and refine their own clusters, by showing them all cluster members. To aid recall 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


