Adapting Spoken Dialog Systems Towards Domains and Users

Ming Sun

CMU-LTI-16-006

Language Technologies Institute
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave., Pittsburgh, PA 15213
www.lti.cs.cmu.edu

Thesis Committee:
Alexander I. Rudnicky, CMU (Chair)
Alan W. Black, CMU
Roni Rosenfeld, CMU
Amanda Stent, YAHOO! Research

Submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
In Language and Information Technologies

Keywords: Lexicon learning, cloud speech recognition adaptation, high-level intention understanding, spoken dialog systems

Abstract

Spoken dialog systems have been widely used, such as the voice applications or agents in smart phone or smart car environments. However, speech systems are built using the developers' understanding of the application domain and of potential users in the field, driven by observations collected by sampling the population at a given time. Therefore, the deployed models may not perfectly fit the real-life usage or may no longer be valid with the changing dynamics of the domain/users over time. A system which automatically adapts to the domain and users is naturally desired. In this thesis, we focus on three realistic problems in language-based communication between human and machine. First, current speech systems with fixed vocabulary have difficulty understanding out-of-vocabulary words (OOVs), leading to misunderstanding or even task failures. Our approaches can learn new words during the conversation or even before the conversation by detecting the presence of OOVs or anticipating the new words ahead of time. Our experiments show that OOV-related recognition and understanding errors can be therefore prevented. Second, cloud-based automatic speech recognition (cloud-ASR) is widely used by current dialog applications. The problem though is that it lacks the flexibility to adapt to domains or users. Our method, which combines hypotheses from 1) a local and adaptive ASR and 2) the cloud-ASR, can provide better recognition accuracy. Third, when interacting with a dialog system, users' intention may go beyond individual domains hosted by the system. However, current multi-domain dialog systems do not have the awareness of the user's high-level intentions, resulting in lost opportunities to assist the user in a timely manner or personalize the interaction experience. We built models to recognize the complex user intentions and enable the system to communicate with the user at the task level, in addition to the individual domain level. We believe that adaptation in these three levels can contribute to the quality of human-machine interactions.

Acknowledgement

First of all, I am grateful to have my four committee members. My advisor Alex guided me for the past 6 years and encouraged me to work on the research I am interested in. He taught me how to shape research ideas and design experiments. He has been extremely supportive to my research ever since — giving me freedom to experiment my ideas, offering helpful advice when I stray off path and even presenting my work at international conferences on my behalf due to travel visa issue. Alan is the one who sparked my curiosity about the speech world when I took his undergraduate course and built a pizza delivery dialog system back in 2009. He has been a fantastic resource for me in brainstorming research ideas, planning my career and telling good jokes. Roni introduced me to statistical language modeling and machine learning. I am honored to have worked with him as teaching assistant in these two courses. His critical thinking and excellent teaching skills (e.g., deriving every formula on board using colorful chalk) equipped me with the necessary techniques to conduct my own research. Amanda has been so supportive and resourceful to me throughout my dissertation. She directed me to all types of inspiring external resources that broadened my horizons. Every time I left a meeting with her, I was able to imagine entirely new research directions which I previously had not thought were possible. I am so lucky to have these four excellent committee members on board throughout my journey of discovering new possibilities in language technologies.

I want to thank Aasish Pappu for playing April fool on me the night before my defense: he tricked me into thinking that it is a US convention to mail a hard copy of the dissertation to my external committee. Alok Parlikar consistently asked me about my defense date almost every other week in the past two years (for some hidden reason). I am fortunate to collaborate with my lab-mates in different projects: Long Qin, Matthew Marge, Yun-Nung Chen, Seshadri Sridharan, Zhenhao Hua and Justin Chiu. It has been a great experience and I have learned a lot from each of them. My work is not possible without the research assistants in my lab: Yulian Tamres-Rudnicky and Arnab Dash. We spent a long but fun time together in the lab collecting data. I also want to thank Thomas Schaaf, Ran Zhao, Jill Lehman and Albert Li for helping me with preparing the defense talk. I want to thank tutors in CMU Global Communication Center for helping me revising the structure and language of this document.

I am grateful to my sponsors — General Motors Advanced Technical Center (Israel) and YAHOO! Research. Especially, I want to thank Dr. Ute Winter. She funded part of my research, brought me to Israel for an internship and since then has been a great resource for me in planning my career.

Most of all, I want to thank my parents and my grandparents for believing in me to complete my PhD. They have to develop so much independence at this age while I am 7000 miles away from home. Without their constant love and understanding, I could not have gone this far. Last but definitely not the least, I want to thank Fei Lu, my fiance and her family for their support. This dissertation is not possible without Fei's constant supervision. Thank you for having faith in me and letting me pursue my dream. Thank you for tolerating me for not only the past two years but also many more to come. This dissertation is as much yours as it is mine.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

The interaction between two parties requires having a shared channel transparent to both participants. In language-based interaction, the factors that may impact the transparency of that channel include, but are not limited to, 1) the capability or skills for both parties to communicate with the same language (understandable vocabulary, accent, etc); 2) the understanding of the goal of the conversation. For example, we can foresee an error-prone conversation if there is a vocabulary mismatch. While one party is not able to recognize the other party's intention for the current conversation, the dialog becomes inefficient. These problems become more significant when one of the two parties is an artificial agent (i.e., conversations between a human user and a spoken dialog system as shown in Fig 1.1). From our daily life, we may experience difficulty switching between two websites providing similar services, partially due to the fact that the vocabulary they use do not transfer to each other with ease (e.g., "origin/destination" in website A vs. "from/to" in B). Also, if the user's intention is too complex to be fully understood by the agent (e.g., "schedule a dinner with friends"), the user may be directed to some irrelevant domain which could lead to failure. Alternatively, the user may have to manually open a series of apps (e.g., SEARCH, OPENTABLE, CALENDER) in order to fulfill that intent. From these examples we can see that, although human users can adjust themselves in many ways to accommodate the agent's incapability, we believe that if the agent can adapt itself toward the user or domain, the amount of effort required from the user side can be reduced and the efficacy of the dialog can be improved.

Adaptive communication systems should adjust dynamically to the peculiarities of interacting with specific users and to reflect the structure of particular domains. Thus, the systems can evolve to better understand users' input and more efficiently generate behavior relevant to the needs of users. Speech understanding provides the front-end for a dialog system and its quality impacts the performance of all subsequent components in the dialog pipeline. Errors generated in speech understanding risk failure in providing utility to the user and require the user to devote time to recovery. Knowing and adapting to the user's language is an important aspect of performance.

Another key aspect of maintaining good communication is being aware of the user's intentions, reflecting their goals, and their preferences in organizing their activities. These factors define the structure of a dialog. Performance can be improved by evolving task models to reflect interaction experience and making appropriate use of an activity's context. Agents that learn about their users could also share useful information and allow a group to benefit from each

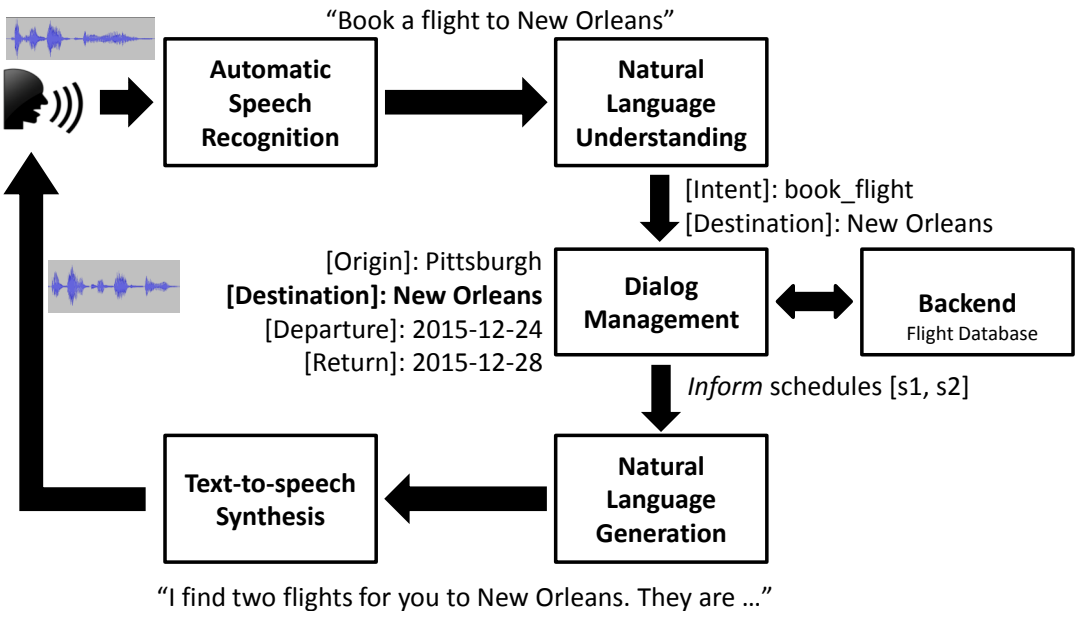


Figure 1.1: Spoken Dialog System

other's experience.

In this thesis work, we mainly focus on better understanding users' language and intentions. First, as the front-end of a dialog system, the spoken language understanding component (including speech recognition) affects the other components in the dialog pipeline. Vocabulary/lexicon is an important part of the understanding model. Vocabulary is time-variant in that more and more new words emerge. Vocabulary is also situated. Depending on who we talk to and what subject we are talking about, the words are different. For example, when discussing some physics problem, the term "electromagnetism" may occur in the conversation. But when talking about sports, words such as "Steelers", "NFL" are more likely. However, it is not always a good idea to incorporate as many vocabularies as possible into the system since improving the coverage of the lexicon may lead to the increase of acoustic confusability which degrades recognition accuracy [62]. Moreover, although the vocabulary coverage is high with large vocabulary ASR (e.g., cloud ASR), there are still domain-specific words such as people's names that cannot be captured. Therefore, an agent which can adjust its vocabulary to learn novel words is useful. Without such capability to dynamically expand/adjust the vocabulary (language model as well), the system is not able to handle the evolving language of users or domains. In chapter 2, we demonstrate two approaches to learning out-of-vocabulary words (OOVs): 1) detect and recover and 2) expect and learn. The former can identify OOVs in the ongoing conversation and recover the spellings of the detected OOVs. Human knowledge can be incorporated as well through conversation. The latter predicts potential OOVs ahead of time and automatically adds those words into the recognition vocabulary.

Dialog system developers either use a local closed-vocabulary speech recognizer or a cloud-based recognizer. Either one has pros and cons. For the closed domain recognizer, although the accuracy on in-domain language is high, the coverage (lexicon-wise and language-wise) is not comparable with the cloud-based recognizer. Nowadays, people tend to use cloud-based

recognizers but the drop of accuracy becomes inevitable in situated dialog systems. However, adaption, which seems to be a reasonable way to improve cloud ASR, is not feasible at this moment due to the lack of such services. In chapter 3, we discuss the work on using a local domain-specific recognizer together with a cloud-based recognizer to make use of the accuracy of local recognizer and the coverage of the cloud-based one. In this thesis, we investigate the form of the local recognizer (finite-state-grammar or statistical language model) and the potential adaptation performance of system combination in terms of reducing word-error-rate (WER).

Dealing with user intentions in the context of individual applications would appear to be tractable, especially as some applications operate in very limited functional domains (for example, a dialer dials; a music player plays music). Contemporary devices, in particular, the ubiquitous smart phone, present the user with suites of applications. Each application will have limited functionality, but they can support fairly complex activities as an ensemble. From human users' perspective, they interact with artificial agents (e.g., smartphone applications) to fulfill some goals. Sometimes these goals can be accomplished via single domain/app (e.g., "find a restaurant" via YELP). However, in many cases, users have more complex goals/intentions that may require coordination across domains. For example, to "organize a dinner event" may involve selecting a restaurant (YELP), contacting friends (MESSENGER) and finding a bus route (MAPS). Without asking a developer to create a single application to solve the "dinner" problem, we would like our agent to automatically learn to perform complex multi-app tasks by using existing apps (i.e., single-task agents). This is more scalable and adaptable to the user's personal needs. However, it requires the agent to be aware of the user's complex multi-app intention so that it understands what resources are necessary to support those goals. In this example, knowing that the user wants to plan a dinner, his preferred dining location and his friend's contacts need to be gathered. Currently, if the user has a multi-app intention, the dialog system does not actively support such interaction unless explicitly configured by the developer. Or, it relies on the user to mentally decompose the task into domains/steps and coordinate information among these domains. Being able to understand user's complex high-level intentions will give the agent an opportunity to construct a specific flow composed of individual domains to accommodate the user's personal needs. This is similar to building a virtual app, mashup, or macro on top of existing functionalities. In Chapter 4 we describe our data collection and modeling process for the agent to learn to recognize users' complex intentions and to assist users at the task level.

1.1 Thesis Statement

In this thesis, we describe techniques that enable a dialog system to accommodate the user's language and intentions. Thus, the system can adapt its vocabulary to better understand the speech input. It is also aware of the user's high-level intentions and can provide assistance at the task level.

1.2 Thesis Contributions

The primary contributions of this thesis are as follows:

- Techniques for improving speech recognition and understanding components in a spoken dialog system by adapting its vocabulary towards the domain and users. By using our first technique, *detect-and-learn*, the dialog system can learn new words during conversation. In Wall Street Journal (WSJ) domain, our model can effectively detect out-of-vocabulary words (OOVs) and reliably learn the correct spellings of those new words, leading to a reduction of the speech recognition word-error-rate (WER). By using our second technique, *expect-and-learn*, the dialog system can harvest potentially useful vocabularies from different resources ahead of time. Experimented on WSJ domain, our system reduces OOV rate by 16.7% and WER by 6.5% in relative by only mildly increasing the vocabulary size. This also leads to improvement of the natural language understanding performance (measured by F_1) at the same time.
- Techniques for adapting cloud-based automatic speech recognition (cloud-ASR) by combining it with local and adaptive recognition. Our hypothesis-selection method chooses one hypothesis from those supplied by cloud and local ASRs. Our experiment on a telephone domain shows that this approach is potentially and practically powerful in reducing WER without heavy feature engineering.
- Techniques for enabling a multi-domain dialog system to understand the user's high-level intentions and actively provide task-level assistance. As a result, the user can address the system with abstract or high-level commands and the system would coordinate several domains to assist the user. Our novel technique, query enrichment, significantly improves the personalized model performance (measured by F_1) by addressing the language mismatch problem. At the same time, the gap between the personalized model and the generic model is significantly reduced after adapting the system output towards specific users. On the other hand, during the conversation, the system can predict the next domain by leveraging the conversational contexts. Our experiment shows that the top-1 prediction accuracy can be improved by 67.7% in relative, compared with the majority baseline with 130 possible apps. In addition to the level of individual domains, the system is also capable of communicating at the level of intentions — it can produce a ranked list of language references to the complex tasks based on user-generated language resources and it can find an acceptable reference within the top-2 of the list.

Chapter 2

Lexicon Adaptation

2.1 Introduction

For spoken dialog systems, vocabulary for a specific domain is often hard to define beforehand, since the vocabulary would evolve over time and the domain data costs time and efforts to collect. People turn to large and generic vocabulary but even that may not well cover the dynamically changing and situated domain language, e.g., person names, landmarks, etc. Therefore, a system which can adapt its vocabulary towards domain or users is more desired.

In this thesis, we designed two approaches to acquiring additional lexicon, similar to how human users naturally acquire new vocabulary in their daily life. Our system can 1) discover unknown words in conversation and recover those words either automatically or by asking the human user for help; 2) expect potentially useful new words beforehand and incorporate them into the lexicon. Two methods have their own use cases. The former learns one word at a time and may involve human knowledge in order to recover words reliably. For example, the user can be asked to provide spelling, meaning of an OOV word. The latter does not have to involve human efforts and can learn a number of words in advance. In this chapter, we discuss the *detect and learn* approach in section 2.2 and the *expect and learn* in section 2.3. Conclusion and future work are provided at the end of this chapter.

2.2 Detect-and-Learn Out-of-vocabulary Words

Most speech recognition systems are closed-vocabulary recognizers and cannot handle out-of-vocabulary. On average, 1 OOV word introduces 1.2 word errors[62]. OOVs are usually content words, like names or landmarks. Therefore, it is important to enable the dialog systems to detect OOVs and incorporate OOVs into the understanding models.

2.2.1 Related Work

People are aware of the out-of-vocabulary problems and investigated ways to handle OOV words in recognition. Hybrid language model with phones, subwords and graphones are adopted [4, 9, 31, 66]. Other information such as confidence score is used to locate possible OOV regions as

well [39, 74, 86]. After detecting OOV regions in an utterance, people use phoneme-to-grapheme alignment to recover the written form of OOV [9, 82].

In this thesis, we adopted a fragment-hybrid language model to detect OOV words during decoding [59, 60]. Different types of fragments are investigated and evaluated in terms of detection and recovery. Based on the work described in this section, detection and recovery of recurrent OOV words can be improved [56, 57]. Language model score can be estimated in order to incorporate the new words into existing understanding models [58]. As a result, 90% of recovered OOV words can be recognized. Moreover, OOVs can be incorporated into other understanding models such as grammar, either by using syntactic or semantic similarities, or by asking human users to provide such information [52].

2.2.2 Method

We trained an open-vocabulary word LM from a large text corpus and a closed-vocabulary fragment LM from the pronunciations of all words in a dictionary. Fragments can be phones, subwords, and graphones etc. When training the word LM, all OOV words were matched to unknown token "$\langle unk \rangle$". Then by combining the word LM and fragment LM, a single fragment-hybrid LM was generated.

As mentioned above, fragments can be phones, subwords, and graphones, etc. Phone and subword only model the phonetic level while graphone considers orthography as well. Such a result of using a fragment-hybrid language model, ideally in-vocabulary words are decoded as words and out-of-vocabulary words are presented as a sequence of fragments. For example, OOV word "ashland" may show up as "AE SH AH N" in a phone-hybrid system, or "EY_SH AH_N" in a subword-hybrid system, or $\binom{ash}{EY\_SH}\binom{en}{AH\_N}$ in a graphone-hybrid system. Note that in these systems, OOVs may not be presented correctly via phone, subword or graphone sequences (as shown in the examples). The goal is to identify such regions and recover from the erroneous presentations.

OOV detection can be further improved by combining three individual systems together via ROVER. The goal here is to more accurately locate OOV word in an utterance, given the OOV reported by three systems. By converting fragment sequences into an "OOV" token, three word hypotheses can be aligned.

To recover OOV words' spellings from recognized fragment sequences, we apply a phoneme-to-grapheme conversion for the phone and subword systems. For graphone system, we simply concatenate the letters together. Other methods such as using search engines (Google, Bing, etc) to auto-correct recovered spellings to further provide more accurate spellings are exploited.

2.2.3 Experiments

We tested our system on the WSJ Nov. 92 5k and 20k evaluation sets [54] using Sphinx3 decoder. The WSJ0 text corpus was used for word LM training. The top 5k and 20k words in this text corpus were used as vocabulary, yielding an OOV rate of 2% for both tasks. Then an open-vocabulary 5k-word LM and 20k-word LM were trained. The dictionary was generated using CMUDict(v.0.7a). The fragment LM was trained from the dictionary. We use WSJ-SI284 as our

acoustic model. We trained bigram fragment-hybrid models for phone, subword and graphone. Word-error-rate using the word bigram LM was 9.23% for 5k task and 12.21% for 20k task.

We reported recall and precision of OOV detection to evaluate individual system's performance (see definition below).

$$Recall = \frac{\text{\#correctly detected OOVs}}{\text{\# OOVs in reference}} \times 100\% \tag{2.1}$$

$$Precision = \frac{\text{\#correctly detected OOVs}}{\text{\# OOVs reported}} \times 100\% \tag{2.2}$$

As shown in the equations, we calculate recall and precision at word level which measures both the presence and positions of OOV words in an utterance since for practical purposes, knowing where OOVs are located in an utterance is more valuable than simply knowing that OOVs exist. Location information can be used later when communicating with humans about OOVs and their meanings since context is available.For OOV recovery, we compared the WER before and after restoring the written forms of OOV words.

2.2.4 Results

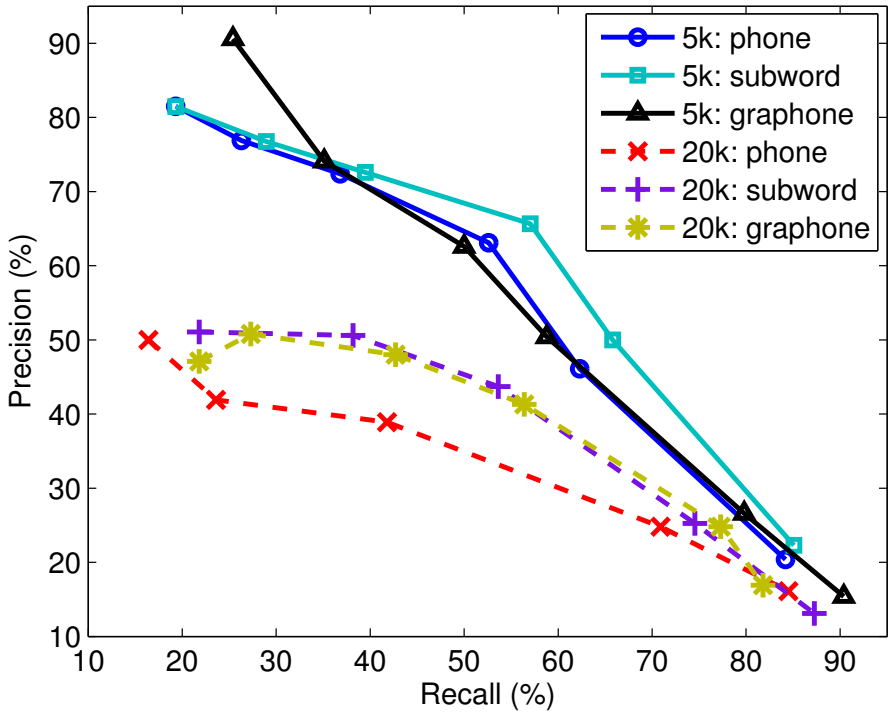


Figure 2.1: OOV detection performance for different fragments

As shown in Fig 2.1, in 5k task, subword outperforms the other two hybrid systems. Both subword and graphone system can utilize longer fragment history than phone system. But graphone system has too many variations even if we constrain the grphone length as short as 2,

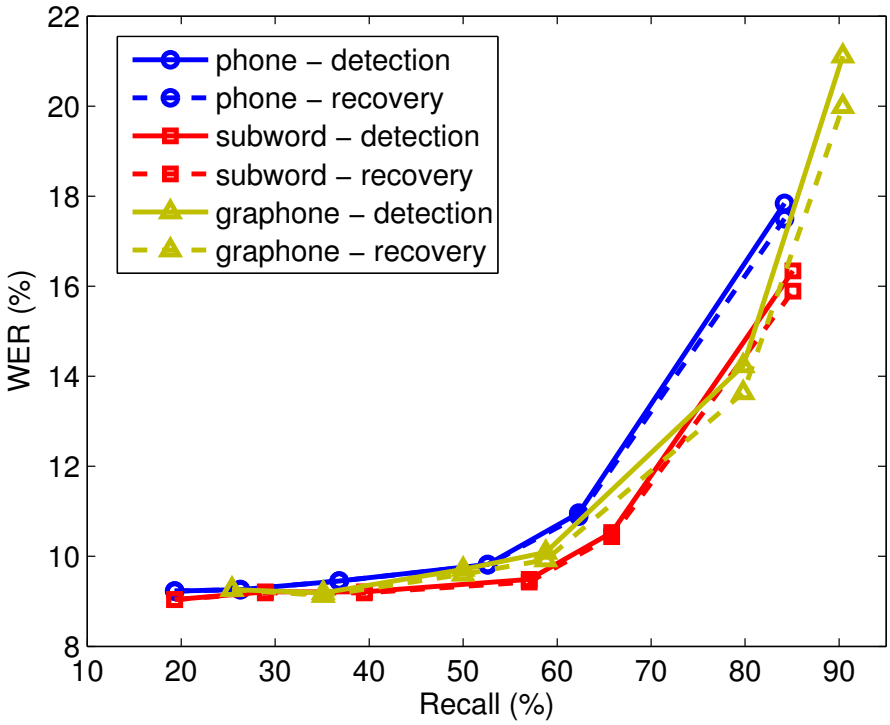


Figure 2.2: OOV recovery results on 5K task

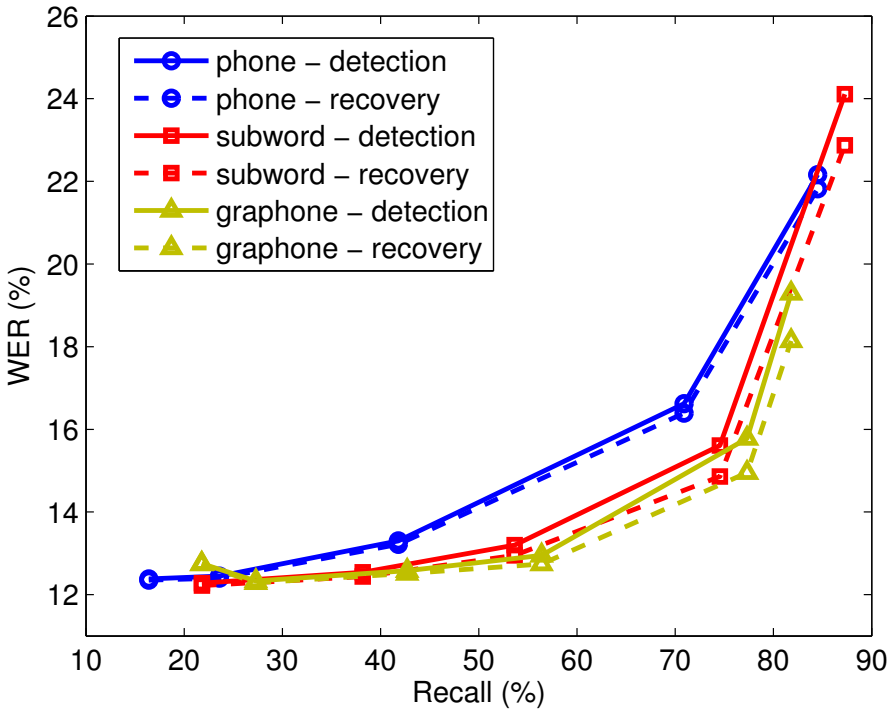


Figure 2.3: OOV recovery results on 20K task

which requires more training data for a reliable language model estimation. Thus, in 20k task, graphone system performance catches up with the subword system and they both are better than phone system.

Fig 2.2 and 2.3 show the OOV recovery performance. Notice that the baseline recognition error without OOV detection and recovery is 9.23% and 12.21% respectively. From the figures, we can see that: 1) OOV recovery improves recognition (compare each solid line with dashed line); 2) subword hybrid system is the best in 5k task and graphone is the best in 20k task (similar to the detection task); 3) in low recall region, system WER can be lower than baseline, which implies that OOVs can be detected with hybrid models without affecting recognition performance.

2.2.5 OOV Learning in Dialog

We integrated OOV detection and recovery into a dialog system and focused on detecting outofvocabulary (OOV) words and learning the correct pronunciation and spelling through conversation. After detecting the presence of an OOV words, the dialog system would encourage the user to say that isolated word once again or put it in a template. As a result, a better phonetic representation would be obtained. Based on this more reliable phone sequence, a preliminary guess of the spelling of the OOV word is generated by a phonemetographeme conversion model and corrected by a spelling correction model. Bing search engine is then used to refine the spelling. After adding the proper pronunciation and spelling into dictionary and LM, the system is able to recognize this OOV word in the next turns. We present a preliminary study as well as the results.

2.2.5.1 User Study 1

The purpose of the experiment is to examine that given the recognized OOV words via a hybrid model, whether or not the system can use dialog strategies to elicit a better representation of the OOV words. We controlled the experiment by forcing the participants to use five sentences from wall street journal data set as the initial input (see Table 2.1). Each sentence contains one OOV word. The OOV detection result shows that four of the five OOVs are recognized with correct boundary and one is partially recognized. For each sentence, we display the text to the participant and asked him/her to read it out, so that the participants knew what they had been talking about. In the system's turn, user's speech was discarded and the original WSJ speech was used together with an OOV detection model. In other word, the system would always find the same OOV region for the same sentence, no matter who is speaking.

To elicit an isolated instance of the OOV word correctly from the user, we designed two types of tasks. First, the system would prompt the participant with a question containing the recognized phone sequence. The candidate questions are shown in Table 2.2. However, since the OOV words are impacted by the context, sometimes the decoder can only produce part of the OOV's real phone sequence. This leads to difficulty for the participants to know which word the system cannot understand indeed. For instance, in the "DEDICATED" example in Table 2.1, only part of the phone sequence of the word is found by the system while the beginning part is recognized as an IV word. Therefore, a recognized OOV should be concatenated with its left or right context (word) to yield better phone sequences, if needed. We adopted a heuristic that if the recognized OOV contains less than 4 phones, the system would apply the context concatenation.

Table 2.1: 5 sentences with OOVs from WSJ

OOV	Sentence	Recognized Phone Sequence
AIRING	As part of the marketing plan the company will begin *AIRING* television commercials during prime time on election night next Tuesday	*EH*R*IH*NG*
DEDICATED	Money managers who sell their firms but then continue working for them may be less *DEDICATED* under new ownership they say	*IH*K*EY*T*AH*D*
SAFRA	The company said its European banking affiliate *SAFRA* republic plans to raise more than four hundred fifty million dollars through an international offering	*S*AA*F*ER*
REFUGEES	Sending *REFUGEES* back isn't their idea it's just what the opposition politicians are saying in our country these days	*R*AH*F*Y*UW*JH*IY*Z*
SALANT	*SALANT* shares closed unchanged on the big board at nine dollars and seventy five cents	*S*EY*L*AH*N*T*

But we found in the pilot study that some participants would answer these question in long sentence, although we specifically asked for an isolated word. This is difficult for the system to generate a precise phonetic representation for the OOV. Even for those who spoken the isolate words, they tend not to speak naturally (e.g., over-articulation issue).

To overcome this problem, we adopted the second task where the participant was asked to use the OOV word in a sentence template when the first answer is much longer than the recognized OOV region. We used "(Please say:) write the word ___ on the paper" as this template. By using this approach, among 7 participants we recruited, 5 succeeded in identifying all 5 OOV words and only 2 failed on "DEDICATED" case. This approach performed better than the first approach so we decided to always ask the template sentence.

2.2.5.2 User Study 2

In the second user study, we integrated this OOV learning into a dialog system. The agent would ask the user to speak about a certain topic. Although our experiment is not topic specific, it helps people to come up with something to talk. This continues until the agent detects an OOV. Then we elicited new instances of the recognized OOV by 1) asking people for the isolated word (Table 2.2) and 2) letting the participant to use the word in a template. As a result, we obtain 3 instances of the same OOV words which allow us to get the spelling. Once recovering the spelling, the agent notifies the user that this word has been learned and encourages the user to speak more about it. Finally the agent would correctly recognized this word as an IV word. In

Table 2.2: Candidate questions to elicit OOVs from user. PHONE represents the recognized fragment sequence.

Questions
There was a word that I didn't understand. It sounded like PHONE . Can you repeat the word once more?
PHONE ? I didn't catch the word. Can you repeat the word once more?
I think I heard something like PHONE . Can you repeat the word once more?
Can you give me one keyword? I guess it sounds like PHONE .

the following, we describe the components in this dialog system.

2.2.5.2.1 ASR and TTS We have three ASR language models to decode three different types of input speech — regular sentence, isolated word and template sentence. To decode a regular speech, we used a subword hybrid LM. The subword LM is trained from WSJ5K dictionary. The subword LM is then merged with a word LM which is trained from WSJ0 text corpus. The total number of subwords we are using is 89 including 39 phones in English and 50 iteratively trained subwords. This setting is reported to perform best for subword hybrid system. To decode a single word utterance, we train a subword trigram model based on the WSJ5K dictionary. To decode the template, we use finite state grammar as our LM. In this LM, only bigrams are modeled. For those transitions between the template words, we set them to 1. For transitions among subwords, we use the bigram LM score in the subword LM. As for the transitions from left context of the OOV into subwords are assigned a probability the same as entering a subword from a startofsentence mark in the subword LM. Similarly, the probabilities for the transitions from subwords into the right context are the same as leaving subwords and entering the endofsentence mark. The acoustic model we are using is si284 model. We observed no much difference by switching to hub4 model.

We used festival with the default voice as our speech synthesizer. For mixed input of text and phoneme sequence we converted it into sable format and used text2wave utility. Before and after every OOV word we inserted a break to make it easier to distinguish.

2.2.5.2.2 Learning OOVs To learn the written form of the OOV word, instead of letting the user spell out the word, we tried to make the system guess the spelling using several resources like phonemetographeme conversion model, spellcorrection model (i.e., Bing search engine). Once initial guess is made by the conversion model which is trained from WSJ5K IV dictionary, we applied a spellcorrection model on top of that. To correct the spelling, we used spell correction service from Bing. Though the exact algorithm is not published it is believed the service is based on not only edit distance but also query similarity to other queries and stored document phrases. We supplied Bing with the initial spelling guess and its surrounding context. For the context we used a window size of 2. We can extract the context words from the user's initial utterance.

However, as mentioned above, in the first run, the context and the OOV phone sequence will influence each other such that neither the OOV nor the context is highly reliable. We proposed a method to address this issue. We temporarily add the phone sequence learned from the template and the initial spelling guess generated by conversion model to the dictionary and LM. Then by redecoding the user's initial utterance which contains the OOV word, the system can get a relatively accurate context.

For simplicity, we assigned a unigram for the OOV and set the LM score and the backoff score high. Alternatively, one can find the class of the OOV and use a classbased LM. Theoretically this should work better than simply assigning a unigram. However, additional conversational turns are required and natural language understanding issues might be involved, which is beyond this course.

2.2.5.2.3 Preliminary Results We recruited two participants (native English speakers). Each of them interacted with the system twice, yielding found dialogs. Two out of these four dialogs were successful — the system learns the OOV and recognizes this word in the future turns. The template-based OOV instance provided reasonable detection of OOV region as well as the fragment sequence. This led to near correct spelling output from the conversion model. For example, in one successful dialog, the conversion model got the exact spelling which was "today" from the initial sentence "I Xeroxed five copies today" by using the template. In this case, the spell correction process would not have further improvement. However, when conversion model produces erroneous spellings, the spell correction is useful. For example, in another successful dialog, the conversion model generated "storrow" from the initial sentence "I am a man of constant sorrow". Even though the spell correction process with only the word as input failed, with the context word "constant", we got the corrected spelling "sorrow".

2.2.6 Detect-and-learn Summary

From the experiments we conducted, we found that OOV detection and recovery is feasible via applying fragment-hybrid language model. Among three individual fragment-hybrid systems, we found that subword and graphone hybrid systems are better than the phone hybrid system for both OOV detection and recovery.

2.3 Expect-and-Learn Out-of-vocabulary Words

Most speech recognition systems use closed-vocabulary [17, 21, 25, 91] and do not accommodate OOVs, leading to recognition errors not only for the OOVs themselves but also the surrounding context words [62]. The OOV-related recognition problem challenges applications such as voice search or spoken dialog systems since OOVs are usually content words such as locations and movie names, which carry the crucial information for the task success [17, 21]. Therefore, a dialog system which can actively learn new words may improve the quality of the conversation.

We want our agent to effectively expand its lexicon guided by in-vocabulary words (IVs) which, by definition, indicates the system's domain of interest to a certain extent. For example, an agent with IVs (or phrases) like *NBA*, *Kobe* and *Chicago Bulls* can probably talk about basketball.

Another agent with IVs such as *availability*, *meeting* and *calendar* can possibly help schedule meetings. However, IVs collected before the deployment of the system may not well cover all possible words in the domain due to aforementioned reasons such as the emergence of new words or the expense of exhaustive domain corpus collection. In this section, we describe our approach to growing the domain vocabulary by incorporating words that are semantically related to the IVs. For example, an IV (e.g., "Celtics") may help the system learn an OOV ("Lakers") if these two words are closely related with each other. We introduce 1) web resources for learning the semantic relatedness among words in Section 2.3.2 and 2) algorithms to expand the vocabulary in Section 2.3.3. Then we describe two experiments in Section 2.3.4 to 1) compare these web resources in terms of their effectiveness in learning new words and 2) demonstrate that with the augmented vocabulary the system performs better in speech recognition and understanding.

2.3.1 Related Work

Out-of-vocabulary words (OOVs) have been studied in the context of speech recognition for a long time. Conventional approaches enable the system to discover the presence of OOVs during conversation [59, 60, 66]. The outcome is that the system would learn one new word at a time. Arguably, approaches of this kind learn OOVs with reasonable precision since the user could a) verify the correctness of the detected OOV region and b) provide additional information to aid the learning (e.g., spelling or an isolated pronunciation of that word). However, large quantity of OOVs may not be learned with limited amount of interactions. Therefore, different from this well-known *detect-and-learn* approach, we enable our system to proactively expect and harvest novel words from web resources ahead of time. We call this less intrusive approach *expect-and-learn*. This method is inspired by [85] where they expand the vocabulary of a parser by adding synonyms of the in-vocabulary words (IVs). We investigate the performance of adding new words into the speech recognizer and we do not restrict new word candidates to just synonyms. For example, given the existing word "dog", in addition to "puppy", we could also learn "cat". We developed two algorithmic heuristics to anticipate new words by leveraging publicly available web resources.

2.3.2 Lexicon Semantic Relatedness

We use semantic relatedness to harvest novel words given a set of IVs. There are multiple ways to measure the semantic relatedness between two words. Linguistic-based semantic relatedness such as WordNet [48] encodes the assumption that words that have similar senses in common are more related to each other. For example, (*knock, punch*) is more related to each other than (*knock, kick*), since "*knock*" and "*punch*" use hand to touch an object while "*kick*" uses foot. Another resource is Paraphrase Database (PPDB) from [23] where two words in language A are considered related if they can be translated to the same word in language B.

On the other hand, the data-driven semantic relatedness approach has been developed to first project a word to a vector space of fixed dimension, i.e., Word2Vec [47] and then directly compute the relatedness between two word vectors. The underlying assumption is that two related words share similar context. For example, (*cat, dog*) is more related than (*cat, turtle*) in the context of "__ *is running in the room*".

2.3.3 OOV Learning Procedure

In this part, we first discuss how to anticipate useful new words based on the training vocabulary. Two algorithms are described in details. We then briefly describe how to add these newly-learned words into recognition models such as lexicon and language model.

2.3.3.1 Anticipating New Words

The first algorithm (see Algorithm 1) learns the OOV words based on the most frequent IV words, where we iteratively extract the most related OOV w^* for each IV word v where v's are ordered by frequency in the training corpus, arguably indicating the importance of each v. On the other hand, the second algorithm (see Algorithm 2) selects OOVs which have high relatedness with all IVs (weighted by IV frequency). In other words, Algorithm 1 focuses on local relatedness while Algorithm 2 cares about global relatedness.

A word relatedness matrix M is built based on the semantic relatedness introduced in Section 2.3.2. Each entry in this matrix represents the similarity between an IV and an OOV: $M_{v,w}$ corresponds to the similarity between the (IV, OOV) pair (v, w). We use $M_{.,w}$ as the vector representation of the similarities between all IVs and an OOV w.

Considering that web resources often have noisy information, the aforementioned data-driven semantic relatedness may generate some noisy words. This issue can be attacked by filtering out garbage words such as mis-spellings or words composed of special characters. In this work, we consult a large external resource to threshold (T) the word frequency for the learned words. We found this helps the agent learn new words with good quality (see Section 2.3.4 for more details).

Algorithm 1 Local OOV Learning Procedure

Require: a set of IV words V, a set of OOV candidates W_v for each $v \in V$, the word relatedness matrix M, a frequency function $f_{\mathcal{D}}(v)$ indicating the word frequency v in domain-specific data $\mathcal{D}$;

Ensure: a set of newly-learned OOV words $W^* \subset W_{v_1} \bigcup W_{v_2} ... \bigcup W_{v_{|V|}}$

Initializing $W^* = \{\}, V^* = \{\}$;
repeat
    $v^* = \arg\max_{v \in \{V - V^*\}} f_{\mathcal{D}}(v)$;
    $w^* = \arg\max_{w \in \{W - W^*\}} M_{v^*,w}$;
    $W^* = W^* + w^*$ and $V^* = V^* + v^*$
until $|W^*| > \theta$
return W^*;

2.3.3.2 Adding New Words to Model

With the two algorithms above, we can learn a list of OOVs which are likely to occur in the future. The current vocabulary can be expanded by adding these learned OOVs. The pronunciation of acquired new words can be learned by looking up a bigger dictionary or applying letter-to-sound conversion model (more details in Section 2.3.4).

Algorithm 2 Global OOV Learning Procedure

Require: a set of IV words V, a set of OOV candidates W, a word relatedness matrix M, a frequency function $f_{\mathcal{D}}(v)$ indicating the word frequency v in domain-specific data $\mathcal{D}$, a frequency vector $\vec{f_{\mathcal{D}}}(V) = [f_{\mathcal{D}}(v_1), ..., f_{\mathcal{D}}(v_{|V|})]^T$;

Ensure: a set of newly-learned OOV words $W^* \subset W$

1: Initializing $W^* = \{\}$;
2: **repeat**
3: $w^* = \arg\max_{w \in \{W - W*\}} M_{.,w} \cdot \vec{f_{\mathcal{D}}}$;
4: $W^* = W^* + w^*$
5: **until** $|W^*| > \theta$
6: **return** W^*;

In addition to expanding the vocabulary, the corresponding language model should be updated to incorporate the OOVs. In this work, we use Kneser-Ney smoothing technique to better estimate the probabilities of the unigrams of the learned OOVs. With the expanded vocabulary and language model, we can perform decoding by using the same acoustic model to evaluate the recognition performance.

2.3.4 Experiments

In this part, we first describe the data sets used in the experiments. Then we discuss two experiments to investigate the *expect-and-learn* approach in the following aspects: 1) the capability of discovering new words and 2) the benefit of learning new words in speech recognition and understanding.

2.3.4.1 Data

To demonstrate the performance of our *expect-and-learn* approach, we examine the results on the Wall Street Journal (WSJ) data set. Since dialog systems are often limited by the insufficient training data, we constrain the training and testing sets in our experiments to be of the same size. This makes the task difficult. In the experiments, we use WSJ 1992 20k-word and 1993 64k-word speaker-independent Eval sets as the test set. We randomly sampled sentences from WSJ SI284 text corpus as our training and development data sets. The sizes (number of sentences) of the training, testing and development sets are 546, 546, 300 sentences respectively. The development set is for tuning the parameters.

In the first experiment, we use three different web resources (PPDB, WordNet and Word2Vec) to generate ranked lists of semantically related words for IVs (e.g., "dog") and harvest new words from the ranked lists. For PPDB, we selected size-L. For WordNet, we compute the path similarity[1] between individual synset in WordNet with each synset of the IV (e.g., there exist multiple entries of "dog" such as 'dog.n.01' and 'chase.v.01' indicating different word senses) and keep the maximum one. For Word2Vec, we used the pre-trained vector representation[2] and

[1] http://www.nltk.org/howto/wordnet.html

[2] GoogleNews-vectors-negative300.bin, https://code.google.com/p/word2vec/

compute the cosine similarity between two words.

When preparing related words for each IV, we constrain the maximum number of related words per IV to be 100 to reduce computation complexity. In the filtering process (introduced in Section 2.3.3.1), we consult the word frequency estimated from a portion (360MB) of 1-billion-word language modeling benchmark data[3] to ensure the quality of the words learned from web resources. We constrain a valid novel word to have occurred at least 1000 times. This could remove 1) rare words and 2) noisy words such as mis-spellings (this is useful for data-driven approach).

In the second experiment, to recognize and understand the speech containing OOVs, we adopt standard WSJ GMM-HMM semi-continuous acoustic model. Word pronunciations are automatically generated by LOGIOS Lexicon Tool[4] based on CMU dictionary[5]. An external vocabulary can be leveraged to compensate the training vocabulary coverage. We use the words extracted from the off-the-shelf US English Generic Language Model[6]. Pocketsphinx is used as our speech recognizer [26]. SEMAFOR, a state-of-the-art frame semantic parser is used for language understanding [18].

2.3.4.2 Experiment 1: OOV Discovery Capability

To compare different learning algorithms with different resources, we measure the OOV rate on the test set as with the training vocabulary augmented by the words learned through the *expect-and-learn* algorithms. Without this learning mechanism, the OOV rate is 22.6%. Three web resources (PPDB, WordNet and Word2Vec) have different characteristics in terms of finding useful new words. As shown in Table 2.3, we can see that first WordNet has the best coverage of IVs. This coverage is important since our algorithms use IVs as seeds to extract related words. Secondly, even though we allow 100 related words to be prepared per IV, WordNet does not find so many words. Path similarity does not exist between some word pairs. This is more severe in PPDB. Word2Vec projects each word into a vector space, allowing computing relatedness between arbitrary word pairs. Third, there is still gap between the rate achieved by the current setup (discussed later in more details) and the oracle OOV rate which incorporates all the related words (100 per IV). Among different web resources, Word2Vec has 1) the lowest OOV rate, 2) the lowest oracle OOV rate and 3) the biggest gap between 1) and 2). The lowest oracle OOV rate may be due to the fact that Word2Vec covers much of the IVs (92.1%) and each IV can generate more related words than the others (97.0 per IV). The remaining gap indicates that useful new words are contained in the related words, requiring a more sophisticated algorithm to be developed in order to mine those words. We show the top related words (phrases) in Table 2.4: Word2Vec can learn more than synonyms. Examples of OOVs salvaged by Word2Vec are shown in Table 2.5.

We implement a baseline which randomly incorporates new words from a (large) domain-independent language model (US English generic language model). We compare our approaches with the baseline. We can use the filtering mechanism mentioned earlier to threshold the min-

[3] http://www.statmt.org/lm-benchmark/1-billion-word-language-modeling-benchmark-r13output.tar.gz

[4] `http://www.speech.cs.cmu.edu/tools/lextool.html`

[5] `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`

[6] `https://sourceforge.net/projects/cmusphinx/files/`

Table 2.3: Oracle *expect-and-learn* OOV discovery capability for each web resource by using training vocabulary alone. External (large) domain-independent dictionary is not used to supplement the domain-vocabulary. Avg. Length: average number of related words per training word. Minimum OOV rate is selected from both algorithms and with thresholding the occurrence of the new words to $T = 1000$.

Resource	IV Coverage	Min. OOV Rate	Oracle OOV Rate	Avg. Length
Word2Vec	92.1%	10.8%	5.1%	97.0
WordNet	100.0%	14.6%	12.9%	77.0
PPDB	80.9%	15.1%	13.5%	2.0

Table 2.4: Examples of the top 5 related words (or phrases) for the word "yellow" with different resources. In WordNet, "yellow" occurs mutliple times due to different senses. In PPDB, "huang" (a common Chinese family name) and "yellow" in English both map to the same Chinese character.

Rank	Word2Vec	WordNet	PPDB
1	red	yellow	yolk
2	bright_yellow	jaundiced	jaune
3	orange	chicken	wong
4	blue	yellow	huang
5	purple	yellow	lutea

imum frequency of learned words, yielding Fig 2.4a and 2.4b. We want to understand the following questions: 1) is there any benefit of using semantic relatedness to harvest new words? 2) is there significant difference between two proposed algorithms? We also discuss other findings such as the utility of the filtering process.

In Fig 2.4, we show the comparison between our approaches and the baselines. Fig 2.4a does not use the word frequency filter while Fig 2.4b does. The baselines (solid lines without markers) mildly reduces the OOV rate, intuitively suggesting that having more words in the vocabulary can prevent OOV occurrence to certain extent. However, note that if these words are noisy, acoustic confusion would be brought to the decoding process.

To address the question 1, we compare our approaches with the baselines in Fig 2.4a and 2.4b. We find that our approaches yield larger OOV rate reduction with less new words. This means that the new words acquired by semantic relatedness are of better quality.

For question 2, we do not see big difference between Algorithm 1 (solid lines with markers) and Algorithm 2 (dashed lines with markers) when filtering process is enabled. In fact, three resources (PPDB, WordNet and Word2Vec) perform similarly at the early stage. But when the filtering is disabled, PPDB benefits from Algorithm 2 while WordNet and Word2Vec works better with Algorithm 1. The difference of OOV rate between two algorithms could be as much as 10% in absolute. Since Algorithm 1 limits the number of new words can be learned per IV, for certain IVs the system cannot find valid new words in their lists of related words — those related words either are IVs themselves or have been learned from previous IVs already. Thus, as we can see

Table 2.5: Examples of OOVs in the test set salvaged by using Word2Vec on source IVs with Algorithms 1. For example, from the IV "American", the algorithm learned the word "British" which occurs in the test set.

Salvaged OOV	Source IV
structure	structures
fell	falling
cost	costs
less	than
British	American
need	should
did	not
above	below
February	month
afternoon	day

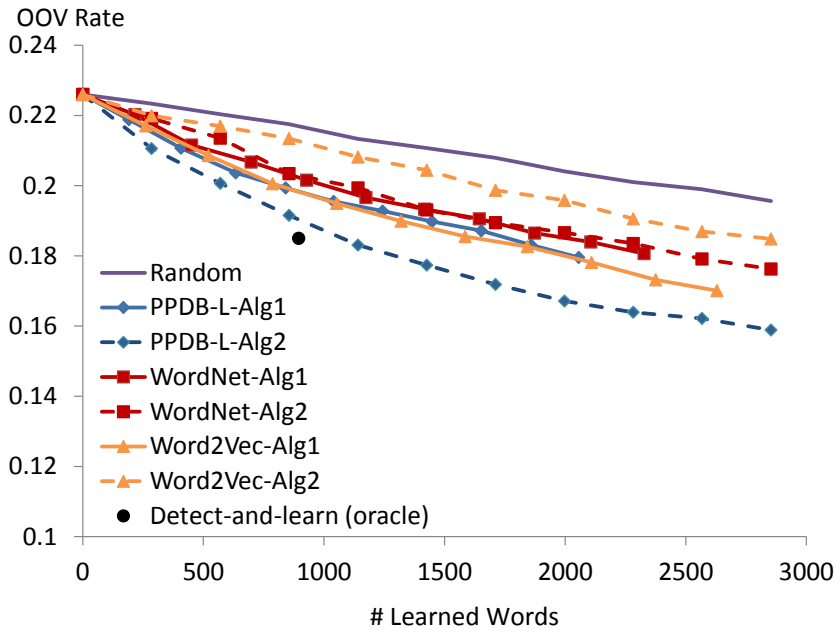
in both Fig 2.4a and Fig 2.4b, the solid lines corresponding to Algorithm 1 are in general shorter than the dashed lines corresponding to Algorithm 2. This would limit Algorithm 1's capability to further reduce OOV rate.

We can see the slopes are much steeper when the filter is enabled. This shows the effectiveness of the filtering process and aligns well with the conventional and intuitive practice that adding frequently used words provides better coverage. However, as we can see from Fig 2.4b, our approaches further improves this intuitive practice. Note that, enabling the filtering mechanism can ensure the quality of learned words, but would restrict the quantity of these words. Compared with Fig 2.4a, PPDB's total number of words generated by both algorithms are reduced by half if the filtering is used. WordNet also suffers mildly. This is probably due to the fact that PPDB and WordNet cannot generate enough words related to IV, compared with Word2Vec (see Table 2.3 for more details).

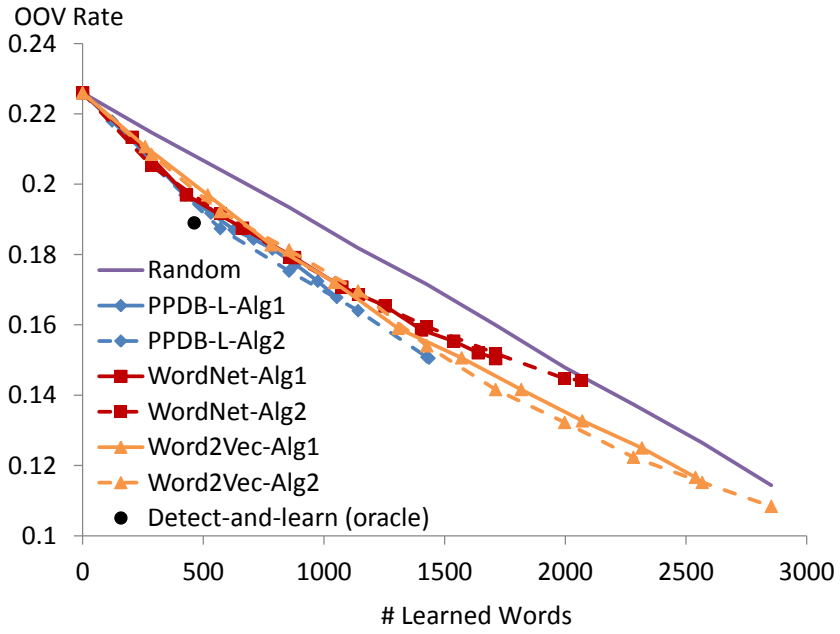
Table 2.6: Breakdown of the test set OOVs. Salvaged OOVs are harvested by using Word2Vec with Algorithm 1 on full training vocabulary. External dictionary is not used to supplement the training vocabulary. Filtering is not enabled. Salvageable OOVs exclude those salvaged ones. Top 10 frequent test set OOVs are shown as examples. "OOV Count" is based on word types.

Condition	OOV Count (pct.)	Examples
Baseline	1399 (100%)	structure, accounting, fell, seoul, ciba, station, cost, research, yuk
Salvaged	513 (37%)	structure, accounting, fell, seoul, cost
Salvageable	536 (38%)	research
Non-salvageable	350 (25%)	ciba, station, yuk

Nevertheless, as we can see from Table 2.3 and 2.6, even if all related words (no more than 100 per IV) in Word2Vec resource are added to the training vocabulary (no additional generic



(a) Without filter



(b) With filter

Figure 2.4: OOV rate drops when new words are learned through different resources. OOVs are harvested on full training vocabulary (2853 entries). For algorithm 1, we allow at most 1 new word to be learned from an IV. For algorithm 2, the maximum number of OOVs allowed to be learned is no more than the number of IVs. For Fig 2.4a, external corpus is not used for thresholding the minimum frequency of the learned words. For Fig 2.4b, external corpus is used and the minimum frequency of the learned words is $T = 1000$.

vocabulary is used), 5.1% of the word tokens (350 word types accounting for 465 tokens) in the test set are still OOVs (i.e., non-salvageable OOVs). Note that although we could have included more related words for each IV, we suspect that those words in the tail of the ranked list may not be useful since the relatedness goes down. Additional external resources may be leveraged such as the aforementioned generic dictionary.

To compare the *expect-and-learn* with the conventional *detect-and-learn* approach, we show isolated dark circles in Fig 2.4, referring to the OOV rate in the test set if *all* OOVs in the development set can be detected and correctly learned, i.e., an oracle upper bound for *detect-and-learn*. As we can see from [59], detection and recovery are not perfect. Although Fig 2.4 shows that the oracle performance of *detect-and-learn* is better than (however close to) our *expect-and-learn* approaches, the number of new words *detect-and-learn* can acquire is limited by the number of OOVs in the previous interactions, not to mention the potential intrusion if the user has to be involved. We believe that *expect-and-learn* complements *detect-and-learn* and both contribute to adapting spoken dialog system's vocabulary.

Table 2.7: The recognition and understanding performance of OOV learning procedures. Word2Vec is used as the resource to harvest new words. Filtering is enabled. Domain (D): training IVs are from domain corpus. Generic+Domain (G+D): training IVs and learned words are further combined with a generic vocabulary.

	Condition	Vocab Size	OOV Rate	Recognition WER	Understanding		
					P	R	F
Domain	(a) Baseline	2854	22.6	49.9	62.6	52.3	57.0
	(b) Alg 1	5394	11.7	41.6	62.4	68.7	65.4
	(c) Alg 2	5394	11.6	42.0	61.8	68.8	65.1
	(d) Oracle	4254	0.0	23.5	81.4	80.5	80.9
Generic+Domain	(e) Baseline	20175	3.6	21.7	80.2	84.4	82.2
	(f) Alg 1	22599	3.0	20.3	81.7	84.8	83.2
	(g) Alg 2	22599	3.0	20.4	81.6	84.9	83.2
	(h) Oracle	20431	0.0	15.1	86.9	87.3	87.1

2.3.4.3 Experiment 2: Benefit of Learning New Words

To investigate the impact of the acquired new words on speech recognition, we compare the word error rate (WER) before and after performing the *expect-and-learn*. There are two baseline setups. First, we constrain the model by building the vocabulary and language model purely based on the training data only (denoted as D). Second, we relax this hard constraint by interpolating the current vocabulary and language model learned on the training data with a generic model (the US English generic language model mentioned above). This is denoted as G+D. On top of both baselines, we use *expect-and-learn* to harvest new vocabulary and adjust the lexicon and language model accordingly. Note that even though G+D has external dictionary, we solely rely on the training vocabulary to learn new words, since the generic dictionary does not tell us the

nature of the domain. In this experiment, we used Word2Vec resource as an illustration. The results are shown in Table 2.7.

The first setup (D) is shown in rows (a)-(d). Row (a) is the baseline result, which only takes domain training data for model training, and performs poorly in recognition due to the limited domain-specific training data. Rows (b) and (c) apply Algorithm 1 and 2 to learn OOVs respectively. It is shown that after learning OOVs, the OOV rates significantly decrease and recognition performance is also improved. Comparing between these two algorithms, their performance is close to each other, which aligns well with the finding from Figure 2.4b. To examine the potential of the OOV learning technique, row (d) shows the oracle results by adding all OOVs in testing data into the vocabulary and language model. The performance can be referred to as the upper bound, where the WER can be decreased from 50% to around 24%, showing the promising potential of OOV learning techniques. However, as indicated in Table 2.3, *expect-and-learn* still has its limitation that not all OOVs can be found from the related words of IVs — resulting in the remaining 5.1% OOV rate if we do not use external resource such as a large generic dictionary.

In the second setup, we interpolate the US English generic language model with the domain language model to analyze the effectiveness of the learning process, as shown in rows (e)-(h). Similarly, it is found that applying the OOV learning approaches improves the OOV rate and the recognition performance compared with the baseline. Also, the oracle result (row (h)) still shows the potential room for improvement. In G+D condition, the US English vocabulary (about 20K words) already covers most of the words in the test set (yielding only 3.6% OOV rate in the baseline). Our learned OOVs, which are outside the generic vocabulary, further improve the vocabulary coverage. As a result, recognition performance is also improved. We believe in a more mismatched situation where dialog system developers have to deal with limited domain data, together with a mismatched generic model, the improvement would be even more noticeable.

In addition to the recognition performance, we also examine the understanding performance after learning OOVs. We conduct semantic parsing on all utterances and extract the outputted semantic frames by SEMAFOR parser. The reference semantic frames are generated by this parser using the manual transcripts. By comparing the outputted semantic frames between the manual transcripts and the decoded results, precision, recall, and F_1 measurements are reported. The results are shown in the last three columns of Table 2.7.

It is obvious that understanding performance can be affected by OOVs. For rows (b) and (c), where a system is built with limited domain data, the understanding performance after learning OOVs becomes better (from 57% to 65% on F_1). The oracle performance achieves even 81% on F_1, showing that it is very important for a system to adapt its vocabulary so as to ensure decent language understanding. The similar trend can be found in the rows (e)-(h). They both suggest that the OOV learning procedure may enable dialog systems to better understand speech input.

2.3.5 Expect-and-Learn Summary

We described our approach to anticipate novel words based on the training vocabulary so as to reduce the occurrence of out-of-vocabulary words. We conducted experiments to investigate and compare different resources and algorithms. We found that having this capability to learn new words ahead of time does reduce OOV rate, yielding better speech recognition and understanding performance.

2.4 Conclusion

In this chapter, we designed and implemented two approaches to learning out-of-vocabulary words. In general, the *detect and learn* and *expect and learn* can effectively reduce the OOV rate and thus improve recognition and understanding performance. *Detect and learn* can be applied during the conversation and can take advantage of human knowledge by using dialog to elicit better pronunciation or orthography of the detected OOV words. *Expect and learn* is an offline learning method and can harvest a batch of semantically related words into the current model. We believe that by adopting these techniques, dialog systems/agents can actively learn novel words and thus effectively preventing OOV-related recognition/understanding errors in the future conversation.

2.5 Possible extensions

2.5.1 Combination of two approaches

Detect and learn should be combined with *expect and learn* in real life, since people naturally apply these two methods to learn new words. *Detect and learn* could provide insight into the weakness or bottleneck of the current lexicon. That would help *expect and learn* to target what words to acquire. For example, during the conversation, if the system detected a few unknown words such as "NFL" or "Steelers", it actually indicates that the agent cannot handle words related to "American football". Given that hint, the agent could expect that users may talk about this sports in the future. Thus, words related with "NFL" or "Steelers" should be harvested.

2.5.2 Topic level similarity in *expect and learn*

Moreover, the agent may need to abstract word-level similarities to topic-level ones. Currently, novel words close to in-vocabulary words are incorporated. In other words, given a set of words, the system would find another set of words. It would be beneficial if the agent can fetch a set of new words given a topic name, e.g., "football" or "food". A use case would be when the user realizes the bottleneck of the system, he could say "learn more words about *football*". This is more feasible and direct than listing a set of words, when the agent would like to involve human knowledge.

2.5.3 OOV learning for dialog system

In this chapter, we demonstrated the feasibility for the agent to learn new words so that it can recognize it from speech input in the future. However, for the rest of the dialog pipeline, it is still an issue to use the newly learned words. As shown in Fig 1.1, suppose "New Orleans" (in red) is the OOV word (phrase), the natural language understanding component needs to understand that it could belong to "destination" category or word class. Thus, the dialog manager can interpret the input and store the extracted information — destination. If this "destination" slot is predefined by the developer, the problem becomes to relate the new word with this existing

slot. Possible techniques include measuring the semantic relatedness between word and slot, where the slot is represented by a set of words collectively, e.g., (“New York’, “Pittsburgh”, “San Francisco”). Consequently, in natural language generation (NLG), by using template-based generation approach, the system just need to use “New Orleans” when the “destination” slot occurs.

However, it is also possible that the newly acquired word does not belong to any of the predefined concepts/slots. Thus, the problem becomes creating a concept for the dialog system on the fly. Using the example above, assuming “destination” is not defined at all, the system needs to create a concept with a label. The label could be arbitrary or meaningless.

Chapter 3

Cloud ASR Adaptation

3.1 Introduction

Large-vocabulary cloud-based speech recognizers are widely used because of their stable performance across environments, domains and users. For example, the Google recognizer [67] is commonly used for Android application development. However, specific domains or users have their own language behavior which cannot be precisely captured by cloud-based recognizers [50]. Intuitively, adaptation to domains, users or environment should improve system performance [6, 34, 84]. In a common use case, smart phones can collect personalized data and use it for adaptation. But such adaptation data do not appear to be practically communicable to a cloud-based speech recognizer. Some companies do not offer this customization service while others do so at a cost that is prohibitive to the individual user.

We propose an alternative scheme that uses a local recognizer capable of dynamic adaptation in combination with cloud ASR. The advantage of doing so is to exploit the domain/user adaptability of the local ASR while using the large coverage provided by cloud ASR. This combination potentially can compensate for the individual weakness of the respective recognizers.

Previous research has examined combining domain-/user- independent Google ASR with domain knowledge to improve recognition by filtering Google results with domain constraints at the word or phonetic level [81]. This approach post-processes Google recognition hypotheses but requires a defined domain and assumes a set of restrictions. Anything beyond this defined domain is not allowed. In addition to the difficulty of precisely defining a domain language, out-of-domain sentences will still be encountered, especially when switching among voice applications via speech. Our proposed method adapts to the domain/user dynamically and out-of-domain hypotheses are not excluded.

In speech recognition, the language model is used to constrain the set of utterances that can be recognized. Two statistically-based paradigms have traditionally been used to derive prior probabilities of a word sequence: finite state grammars and n-gram models [5, 28, 49]. Rule-based finite state grammars are reported to perform better than n-gram models on in-grammar sentences but performs considerably worse on out-of-grammar utterances [32]. Integration of these two models has been investigated previously and has been shown to overcome individual model weakness [5, 45]. In this work, we train probabilistic FSG and SLM models from

the same data and compare their performance given language model adaptation under different circumstances.

Model interpolation can be used for adaptation [6] by training separate models (an adapted one and background one) and then combining the two models. Alternately, an engineer can merge the adaptation corpus and background corpus together and train a single model. However, such background model/corpus is expensive and time-consuming to create and run locally on smart devices. Cache-based language model adaptation has also been used [29]. This method works when the cache size is sufficiently large—implying that a long discourse history is needed. In real life situations, especially in smart device voice applications, interactions with a given application will usually be short in length. And similar to model interpolation, this approach still requires running a large vocabulary recognizer locally on the device.

In the current work, we use collected domain- /speaker- dependent data to directly build a local language model and skip the step of building/adapting a background model. A cloud-based recognizer is used as the background model in combination with local ASR to improve overall recognition performance. We consider language adaptation from three aspects: 1) What form of the local ASR is better suited in a given use case? 2) How much improvement can combining local ASR with cloud-based ASR provide and how can such improvement be achieved? 3) How much data is needed to observe useful adaptation?

The remainder of the chapter is organized as follows. Experiment setup is discussed in Section 3.2. FSG and trigram SLM models are compared in adaptive performance in Section 3.3. Individual model combinations are evaluated in Section 3.4. The necessary amount of adaptation data is assessed in Section 3.5.

3.2 Experiment Setup

We conducted systematic evaluation using data collected from a telephone-based system featuring a small number of calling-oriented applications. A user could ask the system to call a particular individual, call a certain number, redial a previous number, listen to voice mail, etc. In total, 1761 utterances from three native English speakers were collected. Manual transcriptions were made for the material. During data collection, users were shown animations of the tasks to perform, as opposed to written instructions, so as not to restrict the form of their language and to allow for their personal style in specifying names and actions. The number of utterances from each user was 556, 599 and 606 respectively. The total vocabulary size for the 1761 utterances was 195.

Compared with running a large vocabulary recognizer, running locally adapted recognizers is much faster. For example, using the same decoder on a workstation, we achieved 0.35xRT with large vocabulary models (Sphinx US English Generic language model and dictionary[1]). But running with the locally trained recognizer we achieved 0.01-0.02xRT.

We used PocketSphinx0.8 as the local recognizer. A trigram model was trained using the SRILM [73] toolkit, with default settings. A probabilistic FSG was trained using OpenFST [1].

[1] `https://sourceforge.net/projects/cmusphinx/files/Acoustic%20and%20Language%20Models`

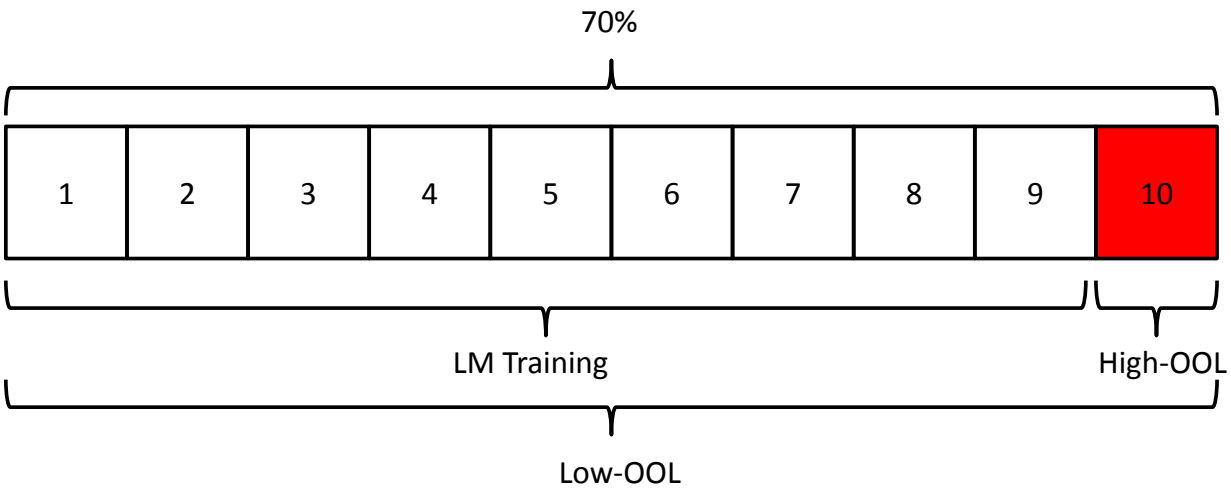


Figure 3.1: Data split on the training set for comparing local recognizers

We used Google ASR[2] as our cloud-based recognizer. We used the same acoustic model for FSG and trigram SLM — the Sphinx US English Generic Acoustic Model. The dictionary was generated using the LOGIOS Toolkit[3] according to the vocabulary of the collected corpus.

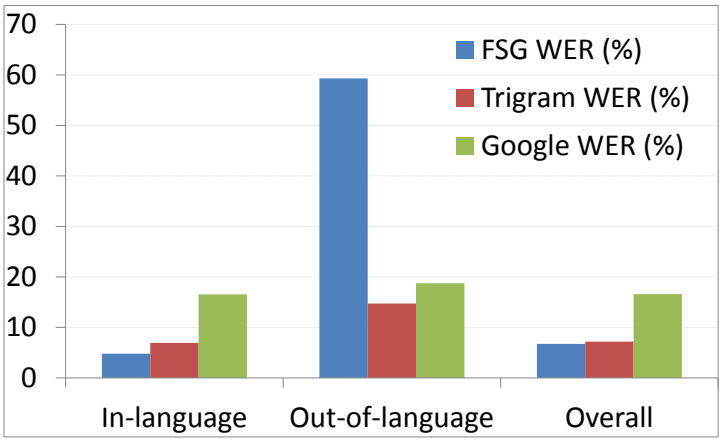
3.3 Local Recognizer Comparison

We ran 10-fold cross-validation on 70% of the 1761 utterances to compare FSG and trigram SLM performance. As illustrated in Fig 3.1, 90% of the full training set was used for training FSG and SLM models and the remaining 10% was for validation. We created two conditions with low and high out-of-language rates (OOL rates) respectively. Here, OOL rate indicates how often the testing sentence has been seen by the model during training. For example, between {"Nice to meet you", "It is nice to meet you"}, if the language model has only seen the sentence "Nice to meet you" during training, the OOL rate is 50%. Therefore, evaluation was performed on these two conditions in each fold respectively. Average word-error-rate (WER) across the 10 folds was reported.

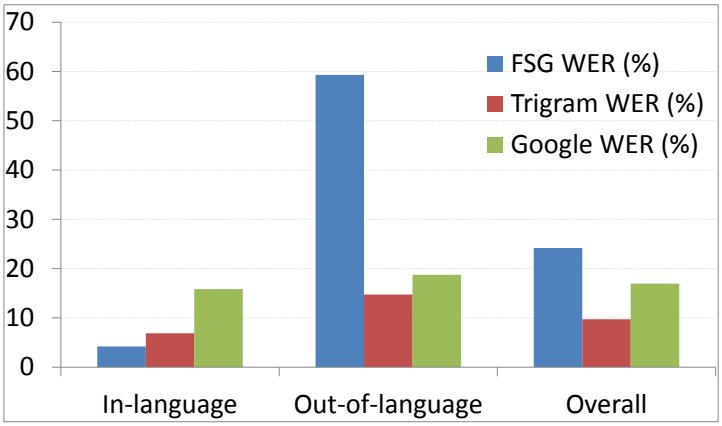
Fig 3.2 shows the performance of each model when the OOL rate is low (Fig 3.2a) or high (Fig 3.2b). We first focus on the "overall" condition, which corresponds to the average WER on the *full* low-OOL and high-OOL data sets in Fig 3.1. We can see that when the OOL rate is low, both FSG and trigram SLM significantly outperform cloud ASR ($p < 0.01$) and FSG is significantly better than trigram. However, when fewer sentences are seen (high-OOL), trigram significantly outperforms the other two ($p < 0.01$). We can see that the performance of both SLM and the cloud ASR is more stable between low and high OOL conditions, while FSG is jeopardised by the OOL sentences.

To confirm the findings above, we then separate the evaluation data into in-language (OOL rate $= 0\%$) and out-of-language (OOL rate $= 100\%$) parts, essentially two extremes for OOL

[2] `http://www.chromium.org/developers/how-tos/api-keys`
[3] `http://svn.code.sf.net/p/cmusphinx/code/trunk/logios/`



(a) Training Set (Average OOL rate = 3%)



(b) Validation Set (Average OOL rate = 34%)

Figure 3.2: 10-fold cross-validation on individual system performance (WER)

rate. From Fig 3.2 we find that for the in-language data, both FSG and trigram SLM perform significantly better than cloud-based ASR ($p < 0.01$). FSG is better than the SLM (significantly better on the full training set). This aligns with the well-known property of FSG that it handles seen sentences well. However, for those out-of-language sentences, FSG performance declines drastically. The domain-specific trigram model is better than the cloud ASR. Observations above indicate that choosing an appropriate local language model may depend on the expected out-of-language rate: domains with very limited language variations may benefit from a FSG model.

Unsurprisingly, we conclude that in a small domain, domain/user dependent local language models will benefit speech recognition accuracy compared with cloud-based recognition. FSG is more suitable when OOL rate is low. Otherwise, trigram SLM is more suitable.

3.4 Combine Local Recognizer with Cloud-based Recognizer

As shown above, local recognizers can improve system performance significantly when adapted to the domain and users. However, by looking into the errors, when OOL rate is high (e.g.,

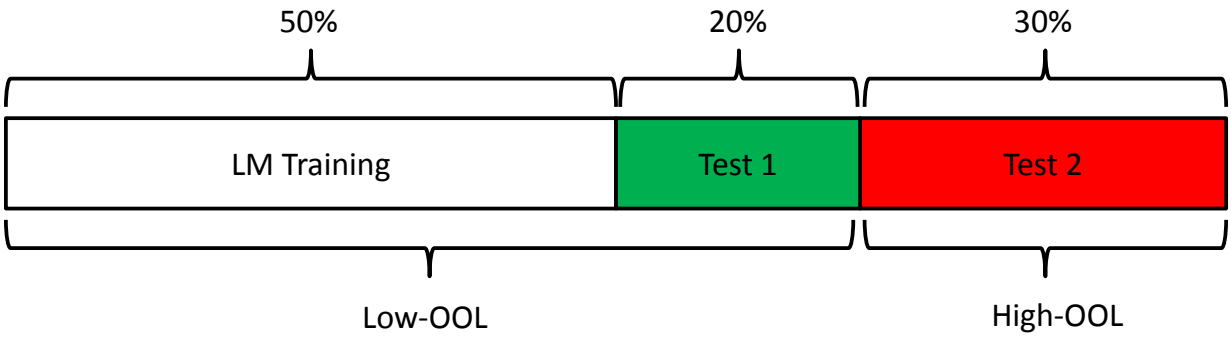


Figure 3.3: Data split for combining local recognizer with cloud recognizer

the validation set in Section 3.3), the majority of the errors (around 80% for FSG and 50% for trigram) are from word sequences not seen in training data. When OOL rate is low (e.g., the training set in Section 3.3), errors related with out-of-language sentences still contribute around 30% of the total errors for FSG. Fortunately, the cloud-based ASR has better performance in this part because of its high coverage: 40% of its errors are from OOL sentences when OOL rate is high and less than 5% of its error are from OOL sentences when OOL rate is low. Therefore, combining the local recognizer with the cloud-based one is intuitively promising.

In this experiment, we combined local recognition with cloud-based recognition to quantify the benefit. Our goals were: 1) to gauge the potential of system combination if a correct hypothesis selection can be made between the local ASR and the cloud-based ASR (an oracle baseline); 2) to build a classifier, using readily available features generated in both recognizers to pick the most likely correct hypothesis.

We segmented the full data into three parts as shown in Fig 3.3: 50% of the data was used for training FSG and trigram SLM. The remaining 50% data was further divided into 20% and 30% for testing. Similar to the practice earlier, we evaluated the trained language models on two conditions: low-OOL and high-OOL conditions. For the low-OOL condition, we combined the LM training data and the 20% portion of the testing data as the evaluation set. For the high-OOL condition, we used the 30% portion of the testing data alone.

Individual system performance is shown in Table 3.1 with WER (%) is shown next to system names. The OOL rate for the low-OOL condition is 11% and 38% for the high-OOL condition. Among individual systems, the trigram adaptive system performs the best. Its WER is 55.4% less than Google WER in relative in the low-OOL condition and 42.6% less than Google WER in the high-OOL condition. FSG performs poorly when the OOL rate is high. Google ASR does not show much difference across between the two conditions (as should be expected).

Table 3.1: Ranked performance of individual systems

Rank	WER (OOL rate $= 11\%$)	WER (OOL rate $= 38\%$)
1	Trigram (7.4)	Trigram (10.1)
2	FSG (11.0)	Google (17.6)
3	Google (16.6)	FSG (27.2)

Table 3.2: Ranked oracle performance of system combinations

Rank	WER (OOL rate = 11%)	WER (OOL rate = 38%)
1	*FSG+Trigram+Google* (2.3)	*FSG+Trigram+Google* (4.4)
2	*FSG+Trigram* (3.3)	*Trigram+Google* (6.0)
3	*FSG+Google* (3.6)	*FSG+Trigram* (6.4)
4	*Trigram+Google* (4.5)	*FSG+Google* (8.6)

We investigate combinations of the three individual systems to evaluate potential improvement. Three two-system combinations (FSG + Google, Trigram + Google, FSG + Trigram) and one three-system combination (FSG + Trigram + Google) are considered. In a real-life use case, depending on the availability of computing power and Internet connection, some individual or combination of the individual systems can be selected according to circumstance.

To combine individual systems, those examples on which recognizers do not agree with each other are labeled as *fsg*, *trigram* or *cloud* depending on which recognizer produces the lowest WER compared with the reference transcription. We use logistic regression from SKLEARN [55] to train a classifier for the two-system combination. For combining three individual systems together, we use K-NEAREST. The classifiers are based on basic features: 1) the number of words in each system's hypothesis; 2) utterance acoustic score (accessible from FSG and SLM systems though not from Google ASR). We show oracle WER in Table 3.2. We also report WER achieved using the basic classifier in Table 3.3. In both tables, systems are ranked according to WER (shown next to system name). Systems which outperform all the individual systems within the combination are italicized.

By comparing the oracle results of system combinations (Table 3.2) and individual systems' performance (Table 3.1), we can see that potentially, system combinations will outperform each individual systems. Combining local FSG, trigram with cloud-based recognition provides the best potential recognition accuracy. From Table 3.2 we can also see: 1) combining more information sources (recognizers) improves the performance; 2) when the OOL rate is low, combinations with local FSG as one component is better than those without FSG since the performance for in-language utterances is most important; 3) when the OOL rate is high, combinations that include the trigram model should be preferred since the adapted trigram demonstrates its advantage in handling both seen and unseen utterances (as described in Section 3.3).

Practically, as shown in Table 3.3, all system combinations achieve lower WER than Google ASR by itself regardless of the amount of OOL utterances. When the OOL rate is low, all combinations beat individual systems comprising the combination. When the OOL rate is high, all combinations show at least 12% relative improvement from Google ASR WER. Trigram combined with Google shows 5% relative improvement compared with trigram alone and 46% less than Google ASR alone.

To summarize, we find that system combinations have the substantial potential in reducing WER compared with individual systems. Combining local ASR (FSG or trigram) with cloud-based ASR can practically (without sophisticated feature engineering) be used to adapt systems to specific domains or speakers without losing the coverage of large vocabulary cloud-based recognizer.

Table 3.3: Ranked performance of system combinations using basic features

Rank	WER (OOL rate = 11%)	WER (OOL rate = 38%)
1	*FSG+Trigram+Google* (3.4)	*Trigram+Google* (9.5)
2	*FSG+Trigram* (5.0)	FSG+Trigram+Google (11.1)
3	*Trigram+Google* (6.5)	FSG+Trigram (11.9)
4	*FSG+Google* (6.9)	*FSG+Google* (15.5)

3.5 Required Data for Adaptation

After observing the benefits of combining local ASR with cloud-based ASR as an adaptation framework, the next question we want to address is how the amount of adaptation data governs improvement. In this experiment, we vary the amount of adaptation data available for building FSG and trigram models and note performance. The ratio between the size of the training set and the size of the testing set is from 0.5:1 to 3:1, reducing the OOL rate from 73% to 29% in the testing set. Recognition performance on the testing set (439 utterances) is reported. Similar to the previous section, we also report the potential improvement (oracle) of combining individual systems. Results are shown in Fig 3.4.

We can observe that when more data is available, individual local systems (dashed lines with markers) perform better and oracle performance of system combinations also improves (solid lines). The two horizontal dashed lines without markers are WER for Google ASR (higher one) and half of its WER as reference baselines. As we can see, for individual systems, FSG WER is approaching Google performance as more training data becomes available. On the other hand, the trigram model beats Google WER even with a small amount of training data (a little more than half of the size of testing data). Moreover, as additional training data is accumulated, trigram performance actually reaches half of the Google WER.

As we can see from the figure, combining trigram with Google (noted as G in the figure) improves faster than the other two combinations (FSG + Google and FSG + trigram). When more data is accumulated, it can reach as low as 30% of Google WER. With training data whose size is only half of the testing data, combining trigram with Google can potentially achieve half of Google WER, while the other two combinations require either equal size of the testing data (FSG + trigram) or more (FSG + Google) to potentially achieve the same performance. Combining three systems together is even more powerful potentially. It can reach as low as 20% of Google WER.

3.6 Conclusion

A real-life limitation of cloud-based speech recognition is the drop in performance observed in application domains that introduce specific vocabulary (for example, personal names) and language use patterns. We investigated a potential solution to this problem that involves supplementing cloud recognition with specialized recognizers that are directly adapted to the domain/speaker. We find that rather modest amounts of adaptation data are sufficient to produce

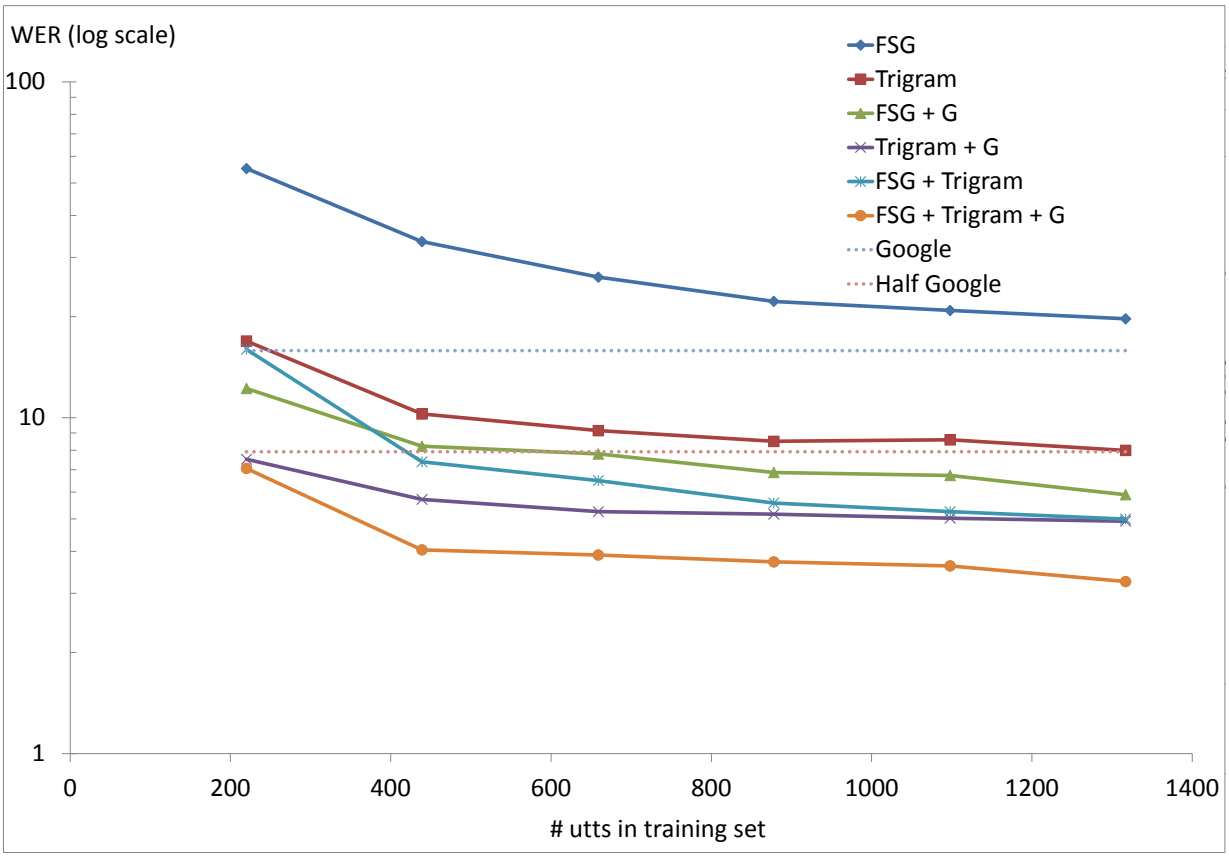


Figure 3.4: System performance (WER in log scale) on different amount of adaptation data

better performance. This suggests that in some use cases, in particular, personal devices with limited computation, a combination of specialized recognition residing on the device with cloud recognition (when available) may show superior performance in comparison with any single recognizer. We should note that the hypothesis-selection classifier we implemented, even in its simple form, leads to better performance. We expect a more systematic investigation of combination strategies will yield even better performance: for example, word-level combinations can produce better performance (as we are observing in other domains).

Chapter 4

Intention Adaptation

4.1 Introduction

Consumers interact with smart devices such as phones, TVs, or cars to fulfill many tasks. Researchers have built numerous academic and commercial systems to handle specific tasks, such as restaurant search [24, 89], event scheduling [53], transit information [61, 75] and so forth. However, a user may use domains such as these collectively to execute higher level intentions: to plan an evening event, for example, requires first finding a restaurant, then booking an event, and, at last, checking the bus schedules to a restaurant or an event afterward. Unless a specific app (dialog agent) is built to handle higher level user intentions, which are themselves comprised of complex combinations of individual domain-specific tasks, the user has to mentally arrange the appropriate set of domains themselves and carry forward information from one domain into subsequent planning stages, across domain boundaries.

Therefore, people have been studying multi-domain dialog systems in the past [13, 14, 38, 43, 51, 63]. These systems host several dialog applications and select a specific one for each speech input. However, such systems are not aware of the user's high-level intentions, nor the potential relationships between different domains. As a consequence, these systems can only passively support cross-domain interactions, i.e., rely on the user to manually transit to the next domain and mentally maintain a shared context across domain boundaries.

Fig 4.1 shows some obvious drawbacks to this approach. First, the agent may miss the opportunity to provide timely assistance to users. For example, if the agent knows that the user actually wants to plan a dinner, it could offer to send messages to friends about the arrangement right after a restaurant is reserved. Second, the conversation between the user and the agent is not efficient. The user may need to input the same information a few times in separate domains. For example, the system first helps the user to find a restaurant. But later, during NAVIGATION domain, it would ask for the destination again since it does not know there exists a relationship between slots in RESTAURANT and NAVIGATION given the current user intention. Redundancies in the process of transferring information from one domain to another are unnecessary. Third, the agent cannot provide the user with any insight into the current state of its understanding about what the user desires in term of the complex intentions. The user would not know if the agent indeed understands what it is asked to do. As dissonance accumulates over time during

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 Täkō 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: Täkō 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 Täkō? 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 4.1: Left: example dialog between user (U) and classic multi-domain dialog system (S); Right: example dialog between user (U) and human assistant (A).

the conversation between the agent and the user, the output becomes increasingly less likely to be what the user needs and errors become more difficult to repair. In short, without knowing the user's high-level intention, the agent will perform poorly.

In response to this, we conducted research to investigate how an intelligent agent can recognize a complex intention from user speech input and provide assistance at the level of intention [77, 78, 79]. Our research demonstrates that our intelligent agent can 1) understand the user's abstract and high-level intention-embedded language; 2) accurately predict the user's follow-up actions; 3) communicate with the user via language at the level of intention, above the level of individual task domains.

In this chapter, we first discuss the related work in Section 4.2. Then our data collection and modeling techniques will be described in Section 4.3 and 4.4, followed by user studies as component-wise and end-to-end evaluation. We focus on 1) high-level intention understanding in Section 4.4.1 and 2) context-based domain prediction in Section 4.4.2. Possible extensions will be provided at the end.

4.2 Related Work

It has been long since the first dialog framework was developed so that domain experts can build their own domain-specific dialog applications [12, 36, 40]. Developers conventionally build dialog apps to handle conversation in specific domains such as restaurant selection, bus

scheduling and navigation [24, 53, 61, 75, 89]. People have been working on extending the dialog systems to handle multiple domains. However, most of the support for transitioning from one app into another and for transporting information from one app into another is handled passively by current technologies, selecting appropriate domains for current user input regardless of centralized or distributed setup [14, 37, 38, 43, 51, 63, 65]. A domain selector (classifier) plays an important role in the spoken language understanding (SLU) component to associate the current speech with one domain. Speech input is passed along to the recognized domain's dialog managemer. As a result, the user still has to verbally initiate/authorize a domain transition. Moreover, the system does not have an understanding of the user's high-level intention, thus missing the opportunity to offer timely assistance.

Besides the conventional domain selection method, some domain is programmed to support transition into another explicitly. For example, although not in the context of spoken dialog systems, YELP (restaurant domain) actually allows users to use MAPS if some navigation information is needed. This approach is promising only if all potential transitions are programmed in advance. This is time-consuming and thus not scalable. Moreover, it cannot adapt to the user's personal needs.

Researchers have found that users conduct a sequence of actions for specific purposes. For example, when browsing the Internet, a sequence of queries (e.g., within a window of time) may contribute to one common interest (information need) [30, 71]. Similarly, we observe that people may use a sequence of apps which handle different domains to accomplish an intention when interacting with smart devices. This will be discussed in more details later.

4.3 Data Collection

We designed a user study to investigate how human users arrange multiple apps together in their daily lives, and also to understand how they would interact with an intelligent agent via speech instead of conventional touch screen. The agent is capable of handling the domains represented by the apps. To collect data for understanding the high-level intentions, we propose a process to: 1) record users' daily app usage by a mobile app; 2) elicit users' knowledge about the nature of those high-level intents; and 3) reenact the intents via speech interactions by a wizard-of-oz system.

In the rest of this section, we first describe the data collection procedure including the logging interface (Section 4.3.1) and user annotation (Section 4.3.2). We then introduce a wizard-of-Oz setup to let the user reenact the interaction through speech in Section 4.3.3. Finally we provide statistics of the corpus we collected in Section 4.3.4.

4.3.1 App Recording Interface

We adopted an Android app[1] that logs each app invocation as an event, together with the date/time and the phone's location (if GPS is available). Episodes were defined as a sequence of app invocations separated by periods of inactivity; based on pilot data we determined that 3 minutes was

[1]This app was developed by Troy Hua and is available at `https://github.com/troyhua/AndroidLogApp`

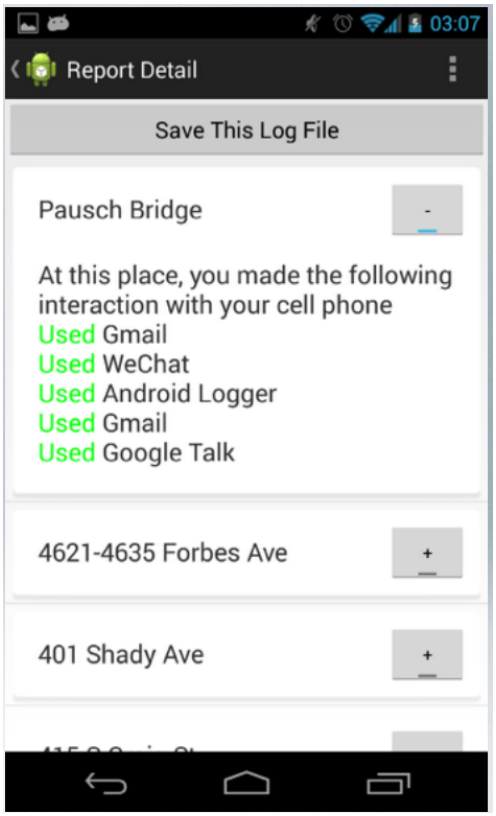


Figure 4.2: Example of activities grouped based on location.

a good minimum duration for episodes in smart phone. Time-based segmentation of a sequence of events is widely used in search query chain analysis [71].

4.3.1.1 Privacy Control

Logs were uploaded by participants on a daily basis, after a privacy step that allowed them to delete episodes that they did not wish to share. As shown in Fig 4.2, activities occurring near each ohter in time were grouped together first. Each such group was represented by an address. The participant could expand a group to see further details such as the apps involved (e.g., GMAIL, WECHAT, etc in Fig 4.2). Participant can swipe this group to remove it from the log if there is any privacy concern. We were informed by participants that they made use of this feature. However, we did not solicit further information about the frequency of use or categories of events. Only information explicitly passed by the user was uploaded.

4.3.2 Task Annotation

Participants were invited to come to our lab on a regular basis (about once a week) to annotate logs and describe the nature of their smart phone activities. Uploaded data was formatted into episodes. Participants were presented with their own episodes with meta-information such as date, time, and location, to aid recall (see Figure 4.4). They were then asked to group events

Figure 4.3: Annotation interface without user annotation

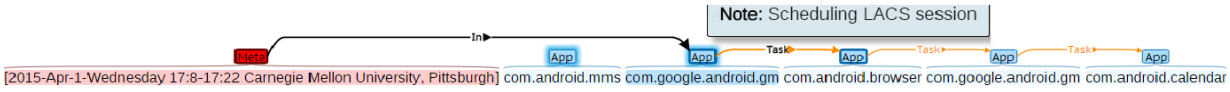


Figure 4.4: Multi-app annotation example; time and location are in red; constituent apps are blue. Users link apps into sequences corresponding to a particular activity (orange link).

(apps) in each episode into high-level intention(s). We observed that episodes could include several unrelated intentions since users could multi-task. Participants were asked to produce two types of annotation, with the brat tool [72] configured for this project. A screen-shot of 4 episodes before user annotation is shown in Fig 4.3.

1. **Task Structure**: link applications that served a common goal/intention.
2. **Task Description**: type in a brief description of the goal or intention of the task.

One example of user annotation is shown in Fig 4.4. The user first linked four apps (GMAIL, BROWSER, GMAIL and CALENDAR) together since they were used for the intention of scheduling a visit to our lab. The user then wrote a description "*scheduling LACS session*". As we observed in the collected data, some of the task descriptions are detailed. In other words, such descriptions themselves propose an app sequence (e.g., "took a picture of my cat and then sent to XXX"). However, many of them are very abstract, such as "look up math problems" or "send picture message".

4.3.3 Task-Related Spoken Dialog

Participants were presented with tasks that they had previously annotated, including the meta-information (date, location and time), their task descriptions, and the apps that had been grouped (Meta, Desc, App lines in Fig 4.5). They were then asked to use a wizard-of-Oz system to perform the task using spoken language. The experiment took place in a lab setting. There was no effort to conceal the wizard arrangement from the participant. An assistant (a 21-year-old male native English speaker) interacted with the participant and was in the same space (albeit not directly visible). The wizard was instructed to respond directly to the participant's goal-directed requests and to not accept out-of-domain inputs. The participants were informed that it was not necessary to follow the order of the applications used on the smart phones. Other than for remaining on-task, we did not constrain their utterances.

The wizard can perform very simple tasks such as "*finding a restaurant using the browser*", "*composing a text message*", "*pointing and shooting a picture*", etc. When asked to take some

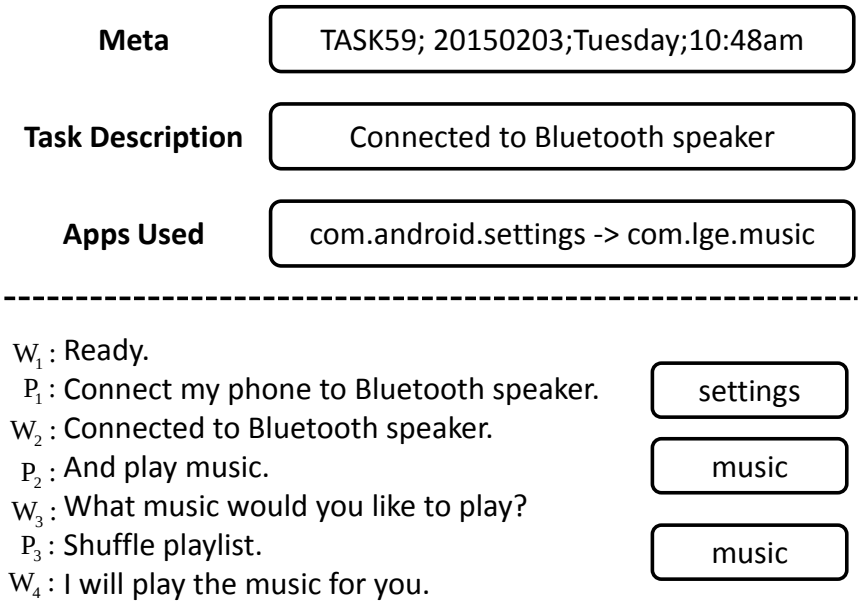


Figure 4.5: Multi-domain dialog example. The top display was shown to the participant; the resulting dialog is shown below.

action, the wizard may request additional information from the participant if necessary according to common sense. For example, the wizard would ask "*what would you like to say in the message?*" or "*which phone would you like to connect to, cell phone or work phone?*". Otherwise, the wizard simply informs the participant of the completion of the task, e.g., "*Ok, I will upload this picture to Facebook*".

Conversations between the participant (P) and the wizard (W) were recorded by Microsoft Kinect device. Recordings were manually segmented into user and wizard utterances. Each utterance is manually transcribed and also decoded by cloud speech recognizer (Google ASR). An example dialog is shown in Fig 4.5.

Each user utterance was later manually associated with the corresponding apps/domains that would handle it. As shown in Fig 4.5, SETTINGS would deal with P_1 to setup a bluetooth connection and MUSIC would take care of P_2 and P_3. However, sometimes users produce utterances which may involve several apps, e.g., "Boost my phone so I can play [game] spiderman" requires CLEANMASTER to clear the RAM and the game SPIDERMAN. Among the total of 1607 utterances, 154 (9.6%) were associated with more than one app — 146 require two apps and 8 require three.

4.3.4 Data Statistics

We recruited 14 participants who already owned Android smartphones, with OS version 4. The participants were recruited via two main channels: 1) flyers on the Carnegie Mellon Pittsburgh campus and 2) the Carnegie Mellon Center for Behavioral and Decision Research[2] (CBDR) participation pool. Table 4.2 provides the demographic breakdown.

[2] `http://cbdr.cmu.edu/`

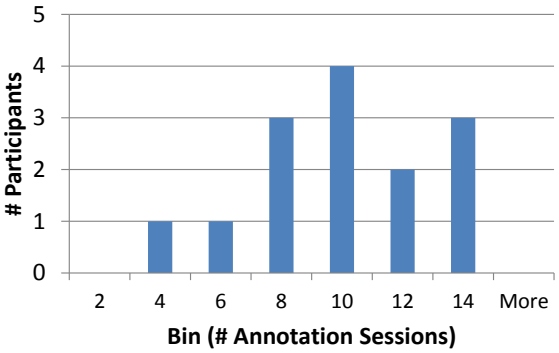


Figure 4.6: Histogram of number of annotation sessions

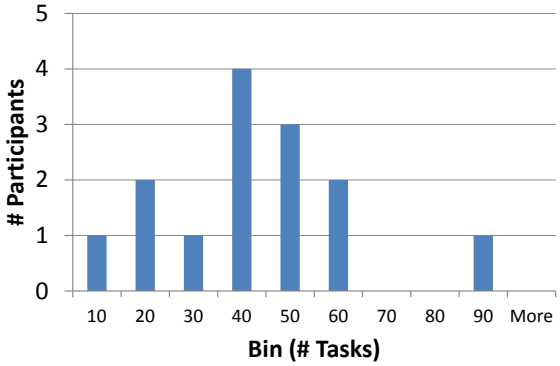


Figure 4.7: Histogram of number of dialogs

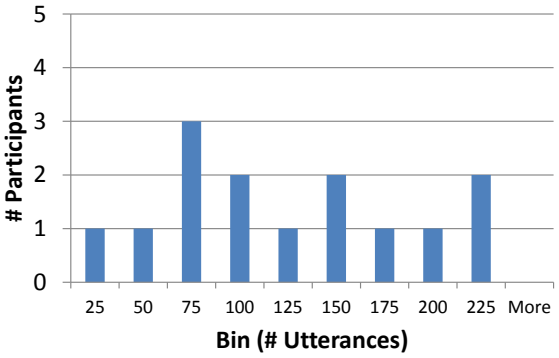


Figure 4.8: Histogram of number of utterances

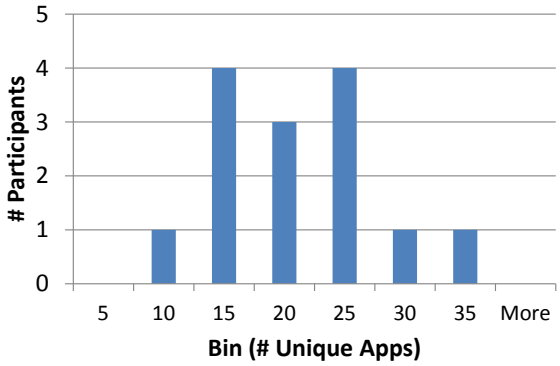


Figure 4.9: Histogram of number of apps

We collected 533 multi-app spoken dialogs with 1607 utterances (on average 3 user utterances per dialog). Among these sessions, we have 455 multi-turn dialogs (involving 2 or more user turns). The breakdown of the 533 dialogs is shown in Table 4.2, where we list the number of participants (#), average age (Age), the number of unique apps involved (#Apps), the number of all dialogues (#Tasks) and multi-turn dialogues (#Multi). As an illustration, we manually cluster part of one participant's multi-domain interactions into groups, based on his commands and the involved apps. As we can see in Table 4.3, this user's language varies even for the same type of tasks. For example, when the user communicates with his family about daily step count, he could say "talk with my family about the step challenge" or "look at my step count then brag to my family". As we will see later, we designed algorithms to overcome address issue.

We used a cloud-based ASR engine (Google ASR) to decode all 1607 utterances and observed word error rate (WER) on the top-1 hypotheses to be 23% (with text normalization). On average, there are 6.6$\pm$4.6 words per user utterance. After removing stop-words[3], there are 4.1$\pm$2.5 words per utterance. The most frequent words across the 14 participants are shown in Table 4.1.

Participants dropped out of the study at different times (see Fig 4.6). On average, each participant annotated 42.4$\pm$21.6 logs during the study. Note that each participant submitted one

[3]`http://www.nltk.org/book/ch02.html`

log per day. In each visit to our lab (less than 1 hour), we asked participants to annotate as much as possible. On average, they annotated 4.3±1.5 logs per visit. Some participants have more than one multi-app task per day while others have less. On average, in our collection, each participant has 1.03±0.70 such tasks per day.

Fig 4.7 and Fig 4.8 show the distribution of number of tasks and utterances over the 14 participants. The correlation between a participant's total number of tasks and the total number of utterances is strong (r=0.92), which is intuitive.

In total, there are 130 unique apps across 14 participants. On average, each user has 19.1±6.1 unique apps. The distribution of the number of apps in shown in Fig 4.9. The correlation between a participant's number of unique apps and number of tasks is moderate (r=0.65). The more multi-app tasks a participant performs, the more unique apps are involved across these tasks.

Table 4.1: Top content words and frequency.

Word	Frequency (%)
open	6.08
text	1.99
go	1.74
please	1.74
send	1.52
picture	1.50
call	1.44
check	1.20
facebook	1.16
message	1.16

Table 4.2: Corpus characteristics. A native Korean and Spanish speaker participated; both are fluent in English.

Category	#	Age	#Apps	#Tasks	#Multi
Male	4	23.0	19.3	42.5	33.3
Female	10	34.6	19.1	36.3	32.2
Age $<$ 25	6	21.2	19.7	44.8	36.3
Age $\geq$ 25	8	38.9	18.8	33.0	29.6
Native	12	31.8	19.3	34.8	28.8
Non-native	2	28.5	18.0	57.5	55.0
Overall	14	31.3	19.1	38.1	32.5

4.4 Modeling User Intentions

People may interact with a multi-domain agent in two ways. First, the user could utter a high-level and abstract instruction/command at the beginning of the interaction such as "plan a dinner

Table 4.3: Manual clustering of one user's multi-domain tasks based on commands and app invocations. Naming of Category is created manually. Contact names are annonymized for privacy concern. Verbs are lemmatized.

Category	Example Commands (count)	Typical Apps
Play games	Allocate memory for spiderman and then play it (2); Play spiderman (2); Play Crossy Road [game] with optimum ram (1); Play and optimize ram for Crossy Road [game] (1); Update and play spiderman (1)	CLEANMASTER, SPIDERMAN, CROSSYROAD, GOOGLEPLAY
Contact people	Text XXX about meeting up (3); Message XXX (2); Text and call XXX (2); Try to contact XXX (1); Ask XXX if she is free to talk via text then call her (1); Look at pictures (1); Download and view a picture XXX sent (1); Message friends (1); Text XXX (1); Text and snapchat with XXX (1); Text friend XXX about going to CVS look up how to get to CVS (1); Send a picture to XXX (1)	MESSENGER, DIALER, SNAPCHAT, CAMERA, MAPS, GALLERY
Do homework	Look up math problems (1); Do physics homework(1); Use calculus software (1); Look up and calculate integrals for calculus (1); Look at homework questions then look up answers online (1)	CAMERA, BROWSER, CALCULATOR, WOLFRAMALPHA
New apps	Download a new game from the app store (2); Look for a song on the play store (1); Listen to and try to buy a new song (1); Purchase Wolfram Alpha on the play store (1)	SHAZAM, GOOGLEPLAY
Share information	View a link XXX sent me (1); Read email and follow link provided in the email (1); Share a link with a friend (1); Make weekend plans with XXX (1); Look at a link XXX sent me and then message her back (1); Look up links to a place my girlfriend and I want to visit (1)	MESSENGER, GMAIL, BROWSER
Exercise	Talk with my family about the (FitBit) step challenge (3); See how much time left I have to win a step challenge on the FitBit app (1); Look at my (FitBit) step count then brag to my family (1);	CLOCK, FITBIT, MESSENGER
Project Communication	Create an email list for a project group (1); Talk with project group (1); Conversation with project group (1); Talk and share with group members (1); Talk with group members about project (1); Email and text group members for a project (1)	GMAIL, MESSENGER

for me". We want our agent to understand this high-level intention and automatically coordinate a set of domains (e.g., RESTAURANT, NAVIGATION, MESSENGER) to assist the user. As a result, the user does not need to mentally coordinate a sequence of domains. This would be especially useful when hands-free interaction is required (e.g., in-car) or for those who have visual disabilities or difficulty with smart devices (e.g., an older population).

The second way in which the user could interact is that s/he may mentally break his intention down into a few steps and communicate each step in order. We want our agent to recognize the high-level intention and provide a smooth transition to the next domain/app. Our model utilizes the current context during a conversation to decide the most probable i) next application and ii) current user intention. For example, when the user requested a restaurant in the afternoon, our model may predict that i) the user may need directions to that restaurant and ii) the user is probably organizing a dinner. Thus, the agent could actively offer the useful information before it is explicitly requested.

In this section, we describe the models suitable for these two use cases. In Section 4.4.1, we introduce an end-to-end framework called HELPR to handle high-level user requests. In Section 4.4.2 we discuss online prediction (i.e., turn-by-turn prediction).

4.4.1 High-level Intentions

We want our agent to help organize apps/domains automatically given user requests expressed at the level of intentions. For example, upon receiving a request like "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 user input, ignoring relationships between domains and the ultimate user intention requiring cross-domain behaviors [13, 14, 15, 16, 37, 38, 43, 51, 63, 65]. This section describes a layer above individual applications that organizes the domain-specific tasks appropriate to overarching user intentions [77, 78]. 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 domain sequences. In the example above, the agent may respond "Okay, to *plan a dinner event*, I need to know *where*, *when* and *who*". Here, by responding with the "*plan a dinner event*" phrase, the agent provides the user with an opportunity to correct potential misunderstanding. *Where*, *when* and *who* collectively construct a shared context across app boundaries. Thus, a unified interaction could be provided, instead of the user having to manage individual domains on their own. This thesis focuses on an agent which is capable of 1) discovering meaningful intentions from user's past interactions; 2) realizing intentions with groups of apps; 3) talking about intentions via natural language. We first describe our HELPR framework and then discuss user studies that evaluate this framework.

4.4.1.1 HELPR Framework

As illustrated in Fig 4.10, the agent maintains an inventory of past interactions, such as "plan a trip to California", each associated with information such as the sequence of involved apps and the user utterances in the (wizard-of-Oz) speech interaction. Given a new input (yellow node),

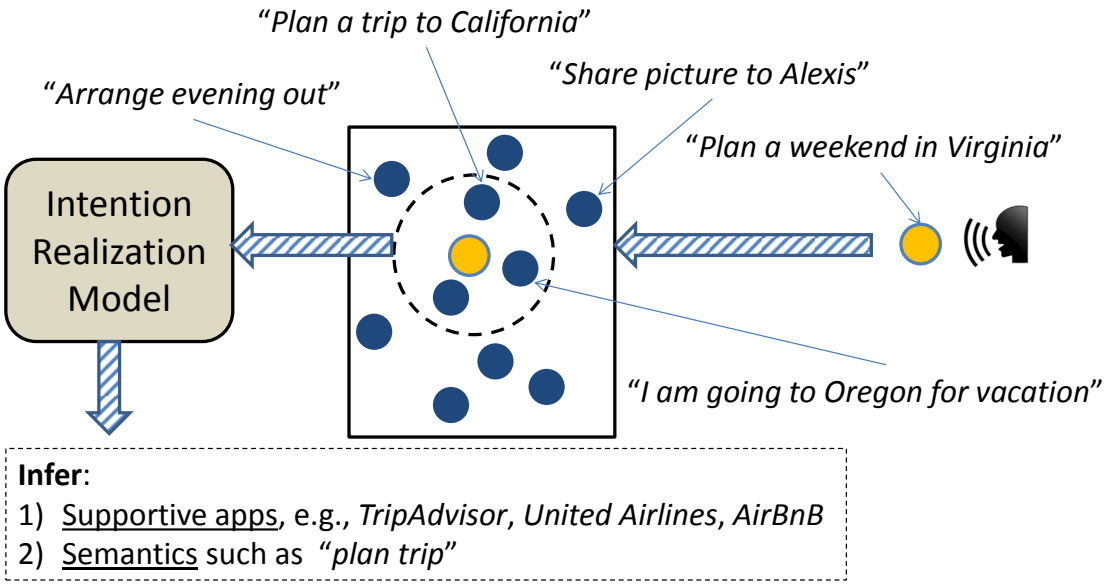


Figure 4.10: Intention understanding and realization example. Solid nodes denote past interactions (blue) and current input (yellow).

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) participating apps and 2) natural language reference. Thus, the intelligent agent transparently conveys its understanding of the input intention in these two modalities.

4.4.1.2 Intention Understanding

We define a complex intention collectively by the set of previous interactions of similar nature. We used two approaches to find similar past experiences. 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. Similar to identifying tasks from search queries for Web Search Engines [42], our goal is to identify basic tasks/intentions from interaction data which is composed of sequence of apps, speech input and a task description. We cluster each participant's data into K_C clusters based on features including 1) apps being used; 2) words in the description; 3) words in user utterances in the dialog.

The group of apps in a multi-app dialog is a natural hint of what the user is trying to achieve, except that the same set of apps may serve different purposes. For example, MAPS can give directions to certain places (navigation task) or provide review information/phone numbers for some business (restaurant reservation task). Using words in user descriptions ("finding a good restaurant") or the actual user utterances ("find me the nearest route to campus") may disambiguate the task identity.

Examples (task descriptions) for clusters in Table 4.4 show that, in general, similar tasks

Table 4.4: Examples of automatic intention clustering of tasks based on utterances, with typical descriptions.

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"

cluster together. To evaluate this cluster-based approach, we asked 6 participants to evaluate the clustering performance on their own data. We showed them the members of each cluster—task descriptions, full dialogs, time and location. For each cluster, we asked them for their agreement with the statement that "the dialogs shown are essentially the same task", from 1 (do not agree at all) to 5 (strongly agree). On average, these 6 participants rated their agreement with 4.2 ± 1.2 out of 5.0. Among the total 51 clusters automatically proposed by the agent, these 6 participants would further divide 10 of them (19.6%) into 27 subgroups. In short, people appear in general satisfied with the clustering algorithm we are using. It is possible that people would manually separate a certain amount of system-proposed clusters each into, on average, 2.7 subgroups. But this interactive process is optional.

In addition to the cluster-based intention model, 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 the cluster-based and the 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 the user to label the current activity while suggesting one of the (ranked) list of basic intentions. (ii) The 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 the neighbor-based method. However, regardless of the differences between these two approaches, we believe that by referring to shared (past) experiences, the agent can (i) better indicate (mis-)understanding of user's intention; and (ii) build rapport with the user [90].

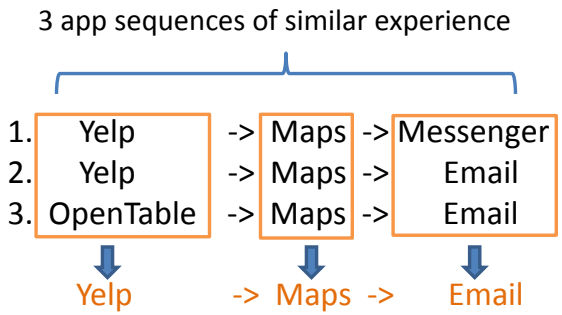


Figure 4.11: Example of RepSeq with three app sequences.

4.4.1.3 Intention Realization in Two Modes

4.4.1.3.1 Participating Applications As an intelligent user interface, the agent needs to assist the 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, as shown in Fig 4.11, three app sequences are first aligned and then a majority vote is performed at each position to generate one sequence. An alternate strategy would be to have a classifier assign multiple labels (app 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.

4.4.1.3.2 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 the user's high-level intention, allowing the user to track the agent's inner state. For example, it can use explicit or implicit confirmation [11], 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"). However, echoing content extracted from the user's current language input does not indicate whether the agent understands or not. Our approach, on the other hand, better communicates the agent's (mis-)understanding by summarizing or abstracting the semantics from similar past experience. This allows timely detection and recovery of errors.

To enable this we want our agent to automatically infer the semantics of the high-level intention from the related past experience. Text summarization can be used to generate a high-level description of the intention cluster [22, 41]. Keyphrase extraction provides an alternative [8, 44, 88]. In our case, we mainly need a short text ("share a picture") so the keyphrase approach is more suitable.

4.4.1.4 Study 1: End-to-End Evaluation

We investigated the differences within: 1) the cluster-based vs. the neighbor-based intention models; 2) personalized vs. generic setups; 3) RepSeq vs. MultLab realization strategies. For each user, the chronologically first 70% of his own collected data was used to train the personalized mode (in principle mirroring actual data accumulation). The other 13 users' first 70% data was combined and used to train the 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 [80]. K_N was set to the square root of the number of training examples [19]. For RepSeq we used ROVER [20] to collapse multiple app sequences into one. For MultLab, we used a support vector machine (SVM) with a linear kernel.

There are intra-user and inter-user inconsistencies in the use of language/apps, creating the problems of *vocabulary mismatch* [42, 69], where interactions related to the same intention may have non-overlapping 1) spoken terms ("take picture" vs. "shoot photo"), sometimes caused by different word selection; 2) app choice, e.g., people may use different apps with essentially similar purposes (MESSENGER vs. EMAIL). To address these issues, we applied query enrichment (QryEnr) and app similarity (AppSim). We describe them in detail.

QryEnr will expand the query by incorporating words semantically close to its words [76], for example $\{shoot, photo\} \rightarrow \{shoot, take, photo, picture, selfie\}$. See Algorithm3 for more details. In short, the chance of observing sparse input feature vectors caused by out-of-vocabulary words (OOVs) is thereby reduced. In this work, we used `word2vec` with the gensim toolkit[4] on the model[5] pre-trained on GoogleNews [47]. Each word w_t in the preprocessed (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 $\vec{f_q}$ [78].

Algorithm 3 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-words feature vector $\vec{f_q} = \{f_1, ..., f_{|V|}\}$ constructed on q; the word semantic relatedness matrix M; the number of semantically similar words N allowed to be extracted for each word in q;

Ensure: an enriched bag-of-words feature vector $\vec{f_q^*} = \{f_1^*, ..., f_{|V|}^*\}$

1: **for** each $w_i \in Q$ **do**
2: Use M to find N words closest to w_i: $V_N = \{v_1, ..., v_N\} \in V$;
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** $\vec{f_q^*}$;

AppSim maps a recommended app, e.g., BROWSER to the preferred (or installed) app on

[4] `https://radimrehurek.com/gensim/`
[5] `https://code.google.com/p/word2vec/`

a specific user's phone, e.g., CHROME. Therefore, similarity metrics among apps are needed to either convert all apps in the generic model training data into the ones that are in this user's phone (as a pre-processing step) or map the recommendation results output by the model to fit this user's preferred (or installed) apps (post-processing step). In the real world, pre-processing 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 the post-processing approach.

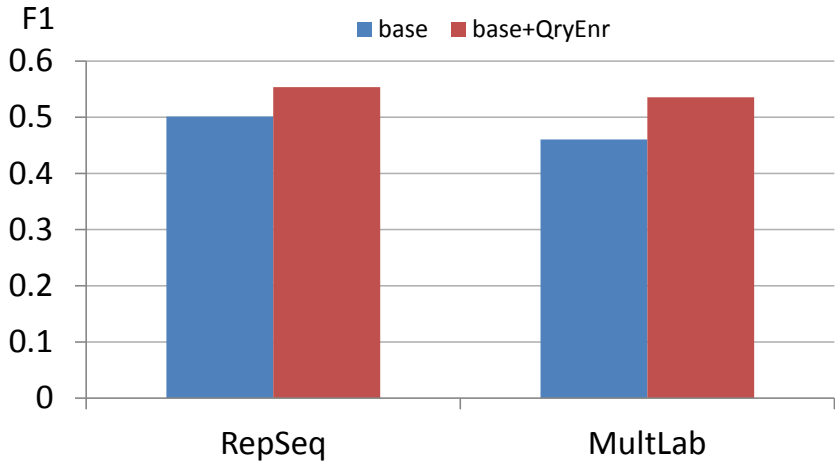
We can construct a similarity matrix among all 130 apps in our collection by three means: (i) rule-based: the app package names can be useful, e.g., com.lge.music is close to com.sec.android.app.music since both contain the string "music"; (ii) knowledge-based: the Google Play store provides a finite ranked list of "similar apps" for each entry; (iii) data-driven: app descriptions from the store can be projected into a low-dimensional semantic space to directly compute similarity. In the rule-based method, we used normalized edit distance with 50 handcrafted 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-driven approach, we used the `doc2vec` toolkit to train the space for over 1 million apps then used cosine similarity [33]. The knowledge-based and data-driven matrices are sparse since some (vendor) apps were not found in our snapshot of the Google database; 15.5% of the 130×130 cells are non-zero for the data-based approach and only 1.0% for the knowledge-based approach.

4.4.1.4.1 Neighbor-based Intention Realization Results We compare the apps suggested by our model with the ones actually launched by users. This prediction task is difficult; in our corpus, on average each user has 19 unique apps and 25 different sequences of apps.

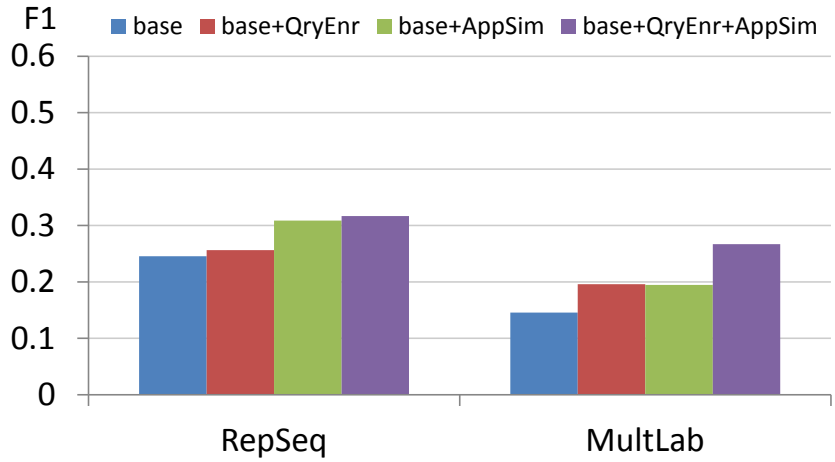
We conduct experiments on the *neighbor*-based intention model. We want to understand the difference 1) between two realization methods — {RepSeq, MultLab}; 2) the effectiveness of the techniques to address language and domain mismatches — {QryEnr, AppSim}. The performance at predicting the participating domains (average F_1 score across 14 participants' testing data) is shown in Fig 4.12. F_1 is higher in Fig 4.12a compared with Fig 4.12b, intuitively suggesting that the personalized model is better than the generic model. We conduct a paired t-test between the base conditions in Fig 4.12a and 4.12b. For RepSeq and MultLab, personalized models are significantly better ($p < 0.01$). The effect size (Cohen's d) is 1.14 and 1.29 respectively.

One can also see that RepSeq outperforms MultLab in both the personalized model and the generic model — the difference is more obvious in the generic model (Fig 4.12b). By conducting a paired t-test, we find significant improvement of the RepSeq base over the MutlLab base ($p < 0.05$) with Cohen's $d = 0.29$, in the personalized setting (Fig 4.12a). In the generic setup (Fig 4.12b), the RepSeq base is significantly better than the MultLab base ($p < 0.01$) with Cohen's $d = 0.62$. This shows that our RepSeq approach can effectively learn people's temporal behavior pattern. It is better than the conventional multi-label classification approach.

Since QryEnr impacts both the personalized and the generic models while AppSim only influences the generic model, we first examine QryEnr in the personalized setup and then investigate QryEnr, AppSim, and their combination in the generic setup. For the personalized model, we conduct a paired t-test between the prediction performance with QryEnr and without



(a) Personalized model



(b) Generic model

Figure 4.12: Evaluation of the neighbor-based intention: Average F_1 on the total 166 testing data across 14 participants. In both figures, base is either RepSeq or MultLab without enabling QryEnr or AppSim. K_N is 18.5 ± 0.4 for generic models and 4.9 ± 1.4 for personalized models. AppSim combines rule-based, knowledge-driven and data-driven approaches with equal weights.

Table 4.5: Examples of intention realization with QryEnr (personalized model). Underlined apps in reference are salvaged.

Command	Reference	Base	Base+QryEnr
Talk to Brooke about step challenge	MESSENGER, FITBIT	MESSENGER	MESSENGER, FITBIT, SNAPCHAT
Turn on WiFi and browse site	SETTINGS, BROWSER, CHROME	BROWSER, BIRDSTEP	SETTINGS, BROWSER, BIRDSTEP
Look at calendar and call Mom	CALENDAR, DIALER	-	DIALER

Table 4.6: Examples of intention realization with AppSim (generic model). Underlined apps in reference are salvaged.

Command	Reference	Base	Base+AppSim
Send picture message	SELFIECAMERA, MESSENGER	ANDROIDCAMERA, MESSENGER	SELFIECAMERA, MESSENGER
Snooze alarm and check notification	CLOCK, WEATHER,GMAIL, FACEBOOK	ANDROIDCLOCK	CLOCK
Add photo Instagram and send picture message	INSTAGRAM, ANDROIDCAMERA, MESSENGER	SNAPCHAT, MESSENGER	INSTAGRAM, MESSENGER

QryEnr (base). For the generic model, we first conduct a correlated one-way ANOVA[6] on {base, base+QryEnr, base+AppSim, base+QryEnr+AppSim} to verify whether there is a significant difference among them. If so, a Tukey HSD test is performed for pair-wise comparisons. In the following, we report our findings in details.

In the personalized models (Fig 4.12a), QryEnr improves the performance at predicting the appropriate set of participating domains. The improvement is significant, measured by the paired t-test for both RepSeq and MultLab bases ($p < 0.05$) with Cohen's $d = 0.41$ and 0.56 respectively. This suggests that individual user's language input is not consistent even for the same task (shown in Table 4.3) and leveraging the semantic relatedness between words can address this issue. Examples in Table 4.5 shows that QryEnr improves the model's ability to recall more domains. However, the drawback is that it may draw forth too many apps.

In the generic models (Fig 4.12b), to examine the usefulness of {QryEnr, AppSim} and their combination, we conduct the ANOVA on the factor with the following four conditions: {base, base+QryEnr, base+AppSim, base+QryEnr+AppSim} for each realization base method. ANOVA finds a significant difference across these conditions ($p < 0.01$). By further adopting the Tukey's HSD test for pair-wise comparisons among the four conditions, we find that QryEnr

[6]Tests are done via `http://vassarstats.net`

Table 4.7: 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 $D = 500$.

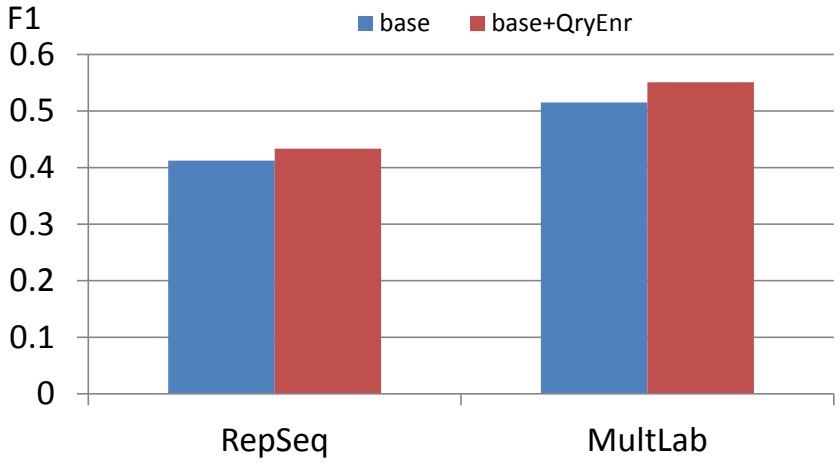
	RepSeq			MultLab		
	Precision	Recall	F_1	Precision	Recall	F_1
Baseline	33.3	18.9	23.8	45.8	12.3	19.1
Rule	43.3	24.3	30.7	59.4	15.9	24.7
Knowledge	41.8	22.3	28.7	53.0	14.6	22.6
Data	38.1	21.2	27.0	54.6	13.9	21.7
Combine	**44.7**	**25.0**	**31.7**	**61.0**	**16.4**	**25.5**

significantly benefits MultLab ($p < 0.05$) but not in RepSeq. One possible reason is that MultLab uses language input twice: it first uses the language command to find similar past experiences and then within that experience cluster, it builds a mapping from commands to apps — errors in finding similar experiences may be recovered. This indeed shows the advantage of the semantic relatedness. On the other hand, AppSim's improvement over the base is more significant in the RepSeq ($p < 0.01$), intuitively suggesting the usefulness of mapping the suggested participating domains to the preferred (or installed) ones. Examples in Table 4.6 shows that the model can effectively personalize the suggestions. In short, in the neighbor-based intention model, QryEnr and AppSim benefit both RepSeq- or MultLab-based generic models in predicting participating apps. While QryEnr works better in the MultLab-based generic model, AppSim works for both RepSeq and MultLab.

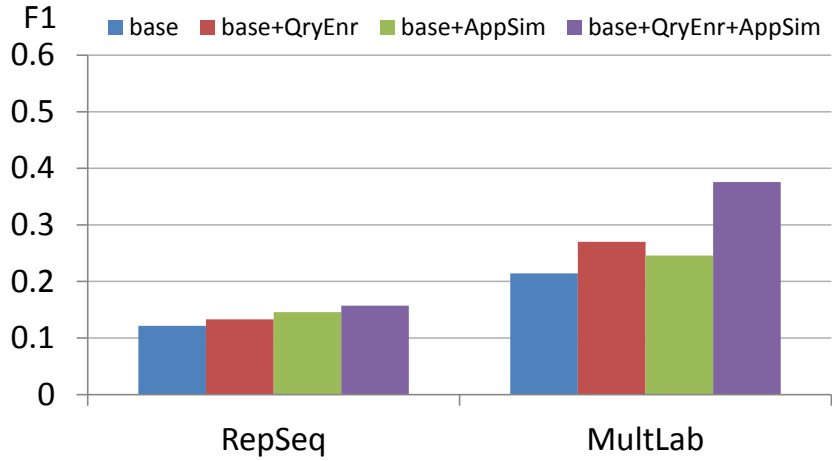
Table 4.7 compares the difference across AppSim methods. The rule-based approach outperforms the other two methods, although it requires filters. This is probably due to the sparseness of similarity matrices in the knowledge-based and the data-driven approaches. Nevertheless, combining three similarity scores yields the best performance, showing the effectiveness of leveraging inter-app similarity for this task.

4.4.1.4.2 Cluster-based Intention Realization Results We conduct the same analysis as Section 4.4.1.4.1. Similarly, as shown in Fig 4.13a, we find the following: 1) QryEnr significantly benefits the personalized models regardless of RepSeq or MultLab ($p < 0.01$). Cohen's $d = 0.40$ for both baselines. This aligns well with the finding in the neighbor-based results. 2) For the generic models (Fig 4.13b), QryEnr significantly improves the base ($p < 0.05$) with MultLab but not with RepSeq. Again, this aligns with neighbor-based results.

There are some major differences from the *neighbor*-based intention models. First, RepSeq in general performs worse with the *cluster*-based intention model (Fig 4.13). The reason could be two-fold: 1) RepSeq relies purely on the selection of cluster members — once the members are fixed, the output participating apps remain the same regardless of the language command. The cluster-based intention model inevitably introduces noise to the process of finding the experience similar to the command. 2) The cluster-based approach may not be ideal when the incoming command is projected close to the intention boundary, suggesting that the current command does not lie close to all the members. However, MultLab could remedy the errors since it leverages



(a) Personalized model



(b) Generic model

Figure 4.13: Evaluation of the cluster-based intention: Average F_1 on the total 166 testing data across 14 participants.

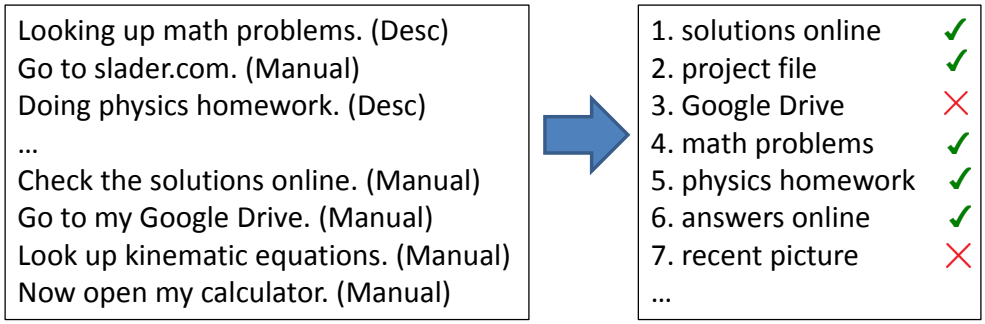


Figure 4.14: Key phrases (ranked) extracted from user-generated language, with user judgment.

the input language to influence the classification output.

The second difference is that we do not see the superior performance of AppSim over the base with the cluster-based intention model. However, as shown in Fig 4.13b, AppSim gains improvement to some extent even though no significant difference is found. Moreover, if AppSim is combined with QryEnr, the system performs significantly better in suggesting appropriate set of apps, compared to QryEnr alone in the MultLab method ($p < 0.01$).

In short, in the cluster-based intention model, QryEnr significantly advances both RepSeq- and MultLab-based personalized models. It also benefits the generic model significantly with MultLab for the same reason as in the neighbor-based intention model. AppSim, on the other hand, shows significant usefulness when it is used together with QryEnr.

4.4.1.5 Study 2: Intention Representation in Natural Language

We used Rapid Automatic Keyword Extraction (RAKE[7]) algorithm [8], an unsupervised, language- and domain-independent extraction method, reported to outperform other unsupervised methods such as TextRank [27, 46] 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 the manual transcription of user utterances from their dialogs (MANUAL), the ASR transcriptions (ASR) thereof and the 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 4.8).

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 a binary judgment for each phrase in the list (randomized) indicating whether it correctly summarized all the activities in the current (refined) cluster. See Fig 4.14 for an example list of key phrases extracted from one user's one intention cluster, along with his binary judgment of individual phrases.

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

Table 4.8: Mean number of phrases generated using different resources

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

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 the 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. Using the more sensitive MAP@K and P@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.

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

4.4.2 Online Prediction

Conventional multi-domain systems do not maintain expectations of any follow-up domains. It is equally likely that the user would talk about food or weather next, regardless of the context. The consequences include 1) the system may not be fully prepared as it could be if it understands how users actually structure such tasks; 2) the system may lose the opportunity to provide timely assistance to smoothly guide the dialog across domains ("Do you want to *send this picture to someone*?") or share the context of the multi-app interaction to the next basic task ("You mean *the picture you just took*?"). In this thesis, we address this issue by conducting a user study investigating how users perform multi-app tasks via language. We demonstrate that systems with shallow understanding, such as what the user said and the common task structures, can still provide improved interaction quality.

It has been shown that based on simple context such as time or location, smart phones that anticipate a user's needs can significantly improve the efficiency of app navigation [70, 83]. However, language interaction may generate more information, allowing for better assistance. In this work, we investigate both conventional features (e.g., previously used apps, time, and location) and language features (e.g., the words in previous user turns) to predict the next domain of interest in the conversation.

4.4.2.1 Context-Aware User Models

Context has been shown to improve the performance of interactive systems [35, 68]. To predict an individual user's next app (e.g., MUSIC) or current intention (e.g., "organize a dinner") during the conversation, we trained personalized user models that include the following contextual

information: 1) **meta context**: time, day of week, location; 2) **behavioral context**: the previously launched app; 3) **language context**: words spoken by the user (e.g., "And play music" in Fig 4.5). These types of context are motivated by our observations; for example: a) people use ALARM more often in the morning on weekdays at home; b) for some user CAMERA is more often followed by MESSENGER to share photos instead of EMAIL; c) "find the address of the Karaoke House in Oakland" not only indicates the use of BROWSER but also hints that the user may want to find the route to the address via MAPS.

However, using content-based *language* features such as words may result in the *vocabulary mismatch* problem [42, 69], where statements related to the same topic may end up with non-overlapping terms, caused by minor differences such as misspellings, morphological differences, synonyms, etc. This can be addressed by enriching the input (user utterances in our case) so similarity measurement can more easily capture semantic relatedness described in Algorithm 3. In the current work we removed stop words and kept only lemmatized[8] verbs and nouns to reduce (morphology) noise.

The proposed system can communicate at the level of low-level actions such as "OK, let me first *find a restaurant in Oakland*" and at the level of high-level intentions such as "I think you want to *plan a dinner*. Let me help you." We implemented this two-level communication by classifying input history to the predicted ranked list of apps or of intentions, respectively. The above three contextual ("meta") features are combined in a bag-of-words. For *time*, we use hours, from the 24-hour clock. For *day*, we use $\{weekday, weekend\}$, which appears to be more informative than $\{Monday, ..., Sunday\}$. For *location*, we use the street name instead of areas based on actual distances if GPS is enabled. We evaluate using top-1 prediction accuracy (ACC) or mean average precision (MAP) over the ranked list of apps reflecting practical use cases.

4.4.2.2 Experiment Setup

In this section, we describe how the agent learns to predict: 1) the next app (low-level); 2) the current user intention (high-level). We also investigate features important for this functionality, and how to combine available features to further improve performance. We introduced approaches to discover basic intentions from the past interactions in Section 4.4.1.2. In this part, we investigate features to recognize the user's high-level intentions during a conversation. We use each user's chronologically first 70% interactions as training data and used the remainder for testing.

4.4.2.3 Predicting the Next App

Results for different features across 14 participants' test data are shown in Table 4.9; we use multi-class logistic regression (individual features are ordered according to MAP on the test set). We use L_2 regularization and use 10-fold cross validation on the individual user's training set to determine the optimal regularization strength ($C \in [0.1, 10]$). The baseline, majority class, is also shown. Note that this is a difficult task; on average each participant has 19 unique apps. As we can see, Last App outperforms Meta features (time, location, day), which others have observed [70]; the Language feature (lemmatized verbs and nouns) is better than Last App.

[8] `http://www.nltk.org/api/nltk.stem.html`

Table 4.9: App prediction

Feature	Train		Test	
	ACC	MAP	ACC	MAP
Language	60.5	66.5	39.1	44.0
Last App	41.9	50.7	29.7	37.2
Time	27.1	36.9	23.3	31.2
Day	26.7	36.2	22.8	30.7
Location	29.9	40.3	21.2	29.4
Majority	*25.8*	*35.2*	*23.9*	*31.7*
Meta	33.2	43.6	20.4	28.4
Meta+App	43.6	52.3	28.0	35.4
Lang+App	**59.2**	**65.3**	**40.0**	**45.0**
All	55.0	61.9	38.8	43.9

Table 4.10: Intention prediction

Feature	Train		Test	
	ACC	MAP	ACC	MAP
Last App	51.9	60.1	52.9	61.7
Language	44.6	53.6	39.3	50.5
Location	40.3	50.4	32.8	44.7
Time	31.5	42.4	31.5	44.4
Day	29.8	40.9	31.0	43.0
Majority	*27.4*	*38.1*	*31.7*	*44.4*
Meta	48.8	58.2	31.7	43.5
Meta+App	58.7	66.3	58.9	66.0
Lang+App	58.9	66.0	54.2	62.7
All	**64.5**	**71.1**	**58.9**	**66.1**

When we combine all individual features (All), we can improve the performance compared to using individual context alone.

Table 4.9 demonstrates that shallow understanding (e.g., what the user said and what app was launched earlier) can be predictive of a user's next actions. For example, given that MAPS is the predicted next app, the GUI can bring it into focus; the assistant can prompt "Would you like to *find the driving directions*?" or even proactively fetch and offer the information, e.g., "By the way, you can *take Bus XXX in five minutes*".

4.4.2.4 Predicting High-Level User Intention

We want the system to be able to 1) discover meaningful intentions and 2) predict the intention for ongoing conversation. We used the clustering process described in Section 4.4.1.2 to identify meaningful basic intentions for each user, based on features such as user-generated language, apps involved and contexts.

Similar to app prediction described earlier, we build a user-dependent multi-class logistic regression model to predict the user intention at each dialog turn. We use data from the 6 users mentioned above, whose clusters were initially proposed by the system automatically and then refined by the user if necessary. We evaluate the performance of each feature in Table 4.10. Again, L_2 regularization is applied with strength $C \in [0.1, 10]$ tuned through 10-fold cross validation. Last App outperforms other features. Best performance is obtained by combining all features. We observe improvement when language is incorporated (All vs. Meta+App), especially in the training set. Careful investigation is needed to understand the lesser improvement in the testing set.

4.5 Conclusion

First, we present a framework, HELPR, that implicitly learns from past interactions to map high-level intentions (e.g., "go out with friends") to specific functionality (apps) available on a smart

device. The proposed agent uses language produced by the 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 may have greater transparency, supporting timely recovery from possible misunderstandings.

Second, we demonstrate that during a conversation, our model can effectively predict the next domain that the user may find useful. This provides the agent insight/expectations to what could happen next. Thus, it can assist the user to smoothly transition to the next domain/app. It is also possible to fetch useful information from the next domain before it is actually requested by the user. As a result, we believe this capability can make the intelligent agent proactively help the user in the conversation.

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 that provided by app-specific interfaces and would allow users to build their own (virtual) applications that combine the functionality of existing apps.

4.6 Possible extensions

The work mentioned in this chapter demonstrates that people have complex tasks (i.e., high-level intentions) to accomplish which span several domains/apps. Our model has been shown to assist users at the task-level. However, there are several remaining challenges in this area. We will describe two of them as follows:

- Efficient accumulation of agent's knowledge of complex tasks from the user's daily life;
- Unified dialog mixed from a set of supportive domains.

There are explicit and implicit ways to acquire knowledge of high-level intentions. First, the user can explicitly instruct the agent by saying "**Watch**! To **plan a trip**, you should *find a cheap flight* from PRICELINE and then *look for a 3 star hotel in downtown* via HOTWIRE ..." Such an instructable agent has been studied in dialog setup [2, 3, 64]. The difficulties lie in the dependency on the agent's capability to comprehend complex language instructions. Alternatively, in a more implicit way, the agent can observe a user perform activities and ask the user for more information if needed. If it does not understand which task category the current activity belongs to, it should request an explanation of the nature of this new task. We focus on the implicit method in this thesis.

The second bottleneck for a multi-domain agent is to produce a unified conversation from the supportive domains. To elaborate, the fundamental communication skills are already provided by domain experts (e.g., "`request destination`" in NAVIGATION domain, "`inform review`" in DINING domain). It is now up to the agent to mix these skills into one conversation. Two obvious issues remain challenging here: 1) The concepts/slots required by different domains may overlap. For example, `destination` in NAVIGATION domain may be implied by the `restaurant_name` in DINING domain. Therefore, it is important for the agent to either pick a subset of concepts or propagate information from observed concepts to unobserved ones, in order

Table 4.11: System actions to acquire new user knowledge based on classification and confidence

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

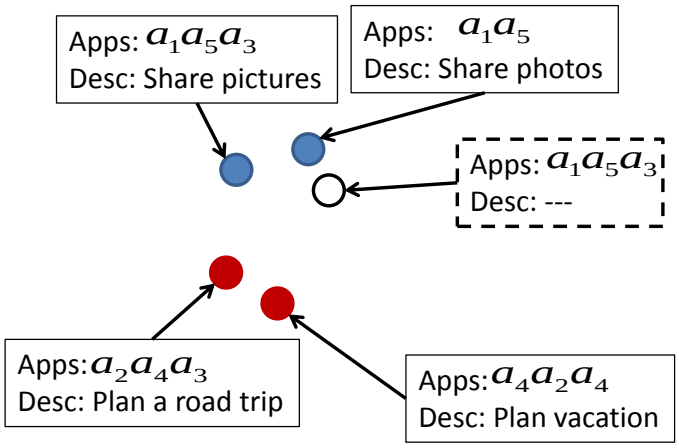
to avoid potential redundancy in conversation. By relating concepts across domain boundaries, the agent can construct and maintain a **shared context**. 2) The dialog flow to request the concepts may be different from concatenating individual domain-specific dialog flows. We will briefly discuss the first challenge. The second one is out of our scope.

In the rest of this section is organized as follows: In Section 4.6.1, we describe our proposed data-gathering approach with preliminary results. Next, we discuss possible solutions to build a shared context in Section 4.6.2. We also discuss a web-based solution to building a multi-domain dialog system, compared with the conventional way of maintaining several dialog managers.

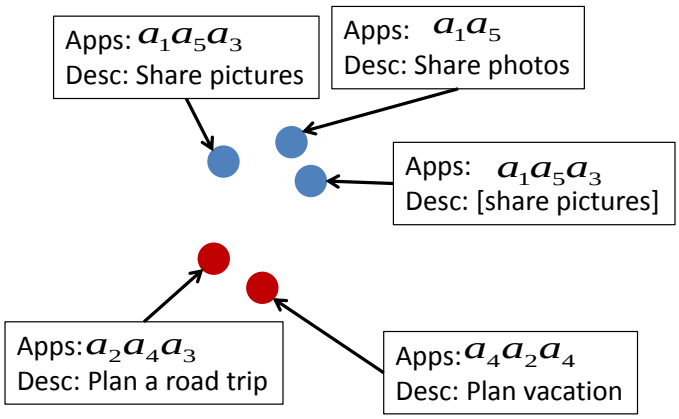
4.6.1 Data Gathering

In our previous work, we asked users to annotate each day's log in terms of what tasks are there and the nature of the tasks. The user would inevitably provide redundant annotation to similar tasks. This would cause degradation of the user experience after the deployment of the agent. It is naturally desired that, after acquiring the user's annotations for a few instances, the agent can by itself decide whether the current task is one of the seen categories of tasks or is a completely new task.

We describe how an intelligent agent may reliably discover and learn complex tasks (i.e., macros) in a realistic setup. An active learning framework may be useful and certain learning mechanisms can facilitate the process. Examples of knowledge acquisition are shown in Fig 4.15 and 4.16. Given the current input, the system may have strong belief that it is a recurrence of a previously seen task. Thus, the agent can directly assign it to that task (see Fig 4.15). On the other hand, the system may find the new observation differs from all past tasks. Thus, it should wait until it observes the new task a few times (see Fig 4.16). Then it could ask the user questions to obtain true nature of the task (Table 4.11).

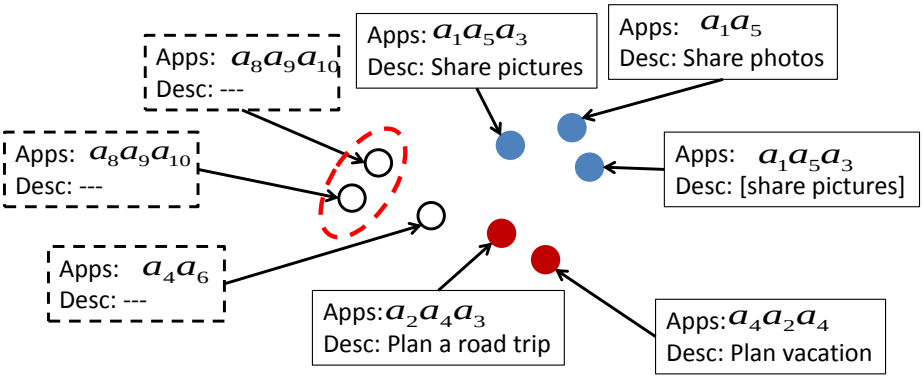


(a) Agent observes new activities

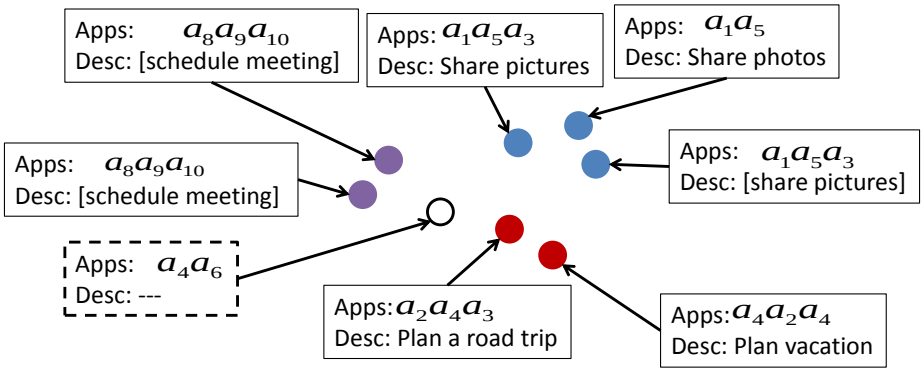


(b) Agent learns the nature of new activities

Figure 4.15: Recurrence: (a) the agent observes a new app sequence (blank node with dashed box). (b) the agent strongly believes that this is similar to one of the previously seen tasks — share pictures.



(a) Agent observes new activities



(b) Agent learns the nature of new activities

Figure 4.16: New task: (a) the agent has 3 observations for which it does not know the task (blank nodes linked with dashed boxes) and 2 of them are similar to each other (within dashed red circle). (b) the agent decides to create a new task for these 2 observations by asking the user a question.

4.6.2 Sharing Context

As mentioned earlier, having a shared context allows the intelligent agent to avoid requesting information it already possesses. Thus, the interaction across multiple domains can be more efficient and natural. We believe this would improve the user experience. The problem of overlapping domain knowledge as well as the targeted shared context are shown in Fig 4.17. In this example, two domains (DINING and NAVIGATION) have a few concepts in common although with different names. When collectively serving a common user intention ("plan a dinner"), knowing the value (or probability over hypotheses) of `Restaurant` slot in DINING domain induces the value (or hypotheses) of `Destination` in the next NAVIGATION domain. A straightforward approach is to activate all domain-specific NLUs and propagate information to the future (denoted as baseline 1). Thus, "Panda Express" (a restaurant chain) would be parsed as both `Restaurant` and `Destination` when talking about DINING and simultaneously filling the `Destination` slot in NAVIGATION. Alternatively, in baseline 2, the agent can look into dialog history and pick up information. For example, it can parse the previously observed user sentences with the currently active (NAVIGATION) NLU and load "Panda Express" to `Destination`, with a decaying function favoring more recent input.

From a dialog state tracking perspective [87], we can formalize this context sharing problem as maintaining and updating the agent's current belief b over all slots including unobserved ones given a newly observed slot. We assume the agent possesses the following knowledge tuple $K = \langle g, D_g, R_g \rangle$: a) user goal/intention g, e.g., "plan a road trip"; b) a finite collection of D_g domains that would assist the current user goal $goal$, where $D_g \geq 2$; and c) the relationship matrix R_g between any two slots from D_g domains — slot_i and slot_j. We anticipate, with appropriate relation matrix R_g and belief update mechanism $b' = f(obs, b, K)$ where obs is the new observation, the context sharing problem can be addressed. Moreover, the methods mentioned earlier (baseline 1&2) are just special cases of this state tracking approach, where $R_g(i, j) = 1$ when input sentence can be parsed into slot_i and slot_j in two domains and 0 otherwise.

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.

4.6.3 Large Scale Dialog Framework with Web Connectivity

Current multi-domain dialog frameworks host several dialog managers from different domains. The advantage of such a framework is effectively distributing responsibilities to domain experts. However, the risk is obvious that the characters or conversational skills are not consistent. Imagine that MAPS talks conservatively (confirming each slot) while YELP talks aggressively (pretending to understand all the user input), the user experience will degrade. Therefore, a centralized agent (CA) may become useful in controlling the conversation quality and user experience. This would implicitly turn each domain into a (web) service, with APIs exposed to the CA. Domain knowledge is now decomposed from a separate dialog manager into a backend supplier. Its role becomes providing/seeking information, rather than deciding the dialog flow. For example, the CA detects the user's dining preference and calls `http://example.com`

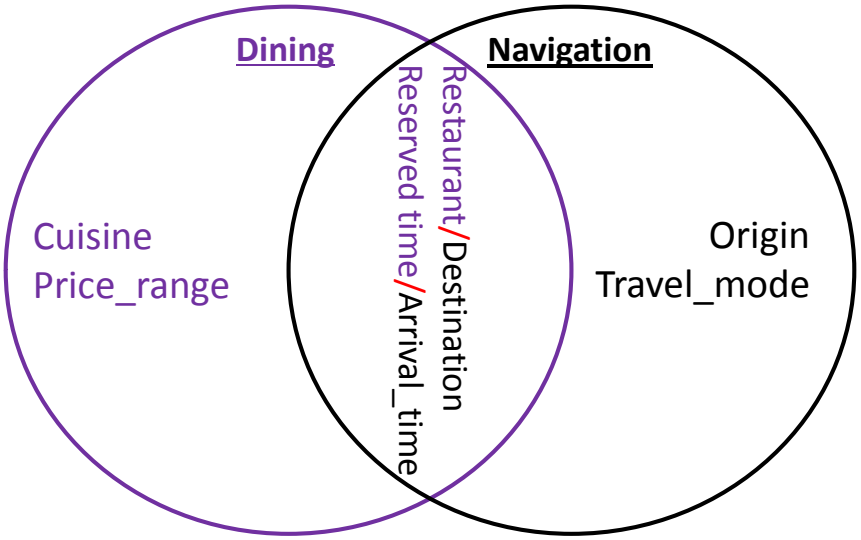


Figure 4.17: Illustration of overlapping domain knowledge.

(or an app) with parameters to suggest a place. This API would generate a structured reply with, for example, restaurant name, address and reviews. The dialog manager would then decide the next dialog act.

The intelligence behind such a framework needs to be able to transfer information between the user and certain APIs. More specifically, it must understand the user's language input and parse it (together with history) into an appropriate service API. On the other hand, it also needs to understand the (structured) reply from a service in terms of the dialog act (inform or request information) and what information to present. For the CA to utilize the resources (domain services), domain developers may use a shared vocabulary to mark up their services (e.g., web pages) with a shared set of schemas such as those from `Schema.org`.

Chapter 5

Conclusion

In this thesis, we investigated adaptation at several levels within spoken dialog systems. We mainly focused on lexicon adaptation, cloud-based ASR adaptation and user intention adaptation in practical use cases, and found such adaptations are possible and practically beneficial. First, our *detect-and-learn* and *expect-and-learn* models can help dialog systems acquire useful novel words such that it can better understand the language in a domain or from a user. Second, by using a device-/domain-specific decoder, together with widely used cloud-based recognition service, the dialog systems can leverage both resources to yield both domain-/user-accuracy as well as coverage. Finally, by being able to understand or adapt to user's personalized high-level intention, the system can learn about people's behavior across domains and learn how to perform complex tasks. This provides intelligent assistance at the task level.

We demonstrated the feasibility and effectiveness of lexicon adaptation, leading to improved recognition and understanding performance. We designed and implemented two strategies to learn new words in real-life situation. First, the system can detect out-of-vocabulary words (OOVs) when talking to a user by using word-fragment hybrid language models. We compared three types of hybrid models to detect the presence of new words in utterances. To learn the spellings of these novel words, we adopted a phoneme-to-grapheme conversion model. Our experiment with the Wall Street Journal dataset demonstrated that the system can reliably detect and learn new words. We also deployed this learning approach in a dialog system to identify new words from real users' speech. Several ways to elicit isolated segmentations of new words from the user were compared and we found that asking the user to put the new word in a template may be better than simply having the user repeat the new word. Our user study result shows that it is feasible for an agent to acquire new words through conversation.

Second, a number of potentially useful new words can be learned beforehand such that when mentioned in the user's utterances later, these words are no longer new words to the agent. Our approach acquired new words that are semantically related to the current vocabulary and put them into recognition models. We experimented with different web resources and found that word2vec outperforms others in terms of the quality of the harvested new words. In short, knowing the current lexicon gives the agent useful directions to explore unknown words, leading to improvement in speech recognition and understanding.

To tackle the current impossibility in adapting widely used domain-/user-independent cloud speech recognizers, we investigated the benefit of combining local ASR with cloud ASR to marry

each individual system's strength, namely 1) domain-/user-adaptability with local ASR and 2) stable performance across domains and users with cloud ASR. Local ASR provides in-domain accuracy while cloud ASR has out-of-domain coverage. We implemented two local ASR models and found that in different use cases, a finite-state-grammar based language model and an n-gram model have their own advantages and disadvantages. We implemented a hypothesis selection model to choose the appropriate decoding result from an ensemble of decoders, depending on the availability of resources such as Internet or computing power. We found that this ensemble of individual local decoders with cloud one can adapt the recognition towards domains or users and thus yield better accuracy.

Intelligent agents are developed to assist a human user with tasks. As a consequence of the users' high-level intentions that may go beyond the individual domains, users inevitably have to make an effort to maintain contexts across domain boundaries or initiate transitions between domains. Therefore, it is desired that the agent can create macros or virtual apps based on existing functionalities to assist users with complex tasks. We conducted a user study to collect real-life multi-domain tasks from real users. The data is composed of two modalities — touchscreen interactions on smart phones and spoken dialogs with a wizard-of-Oz agent. We implemented user models to understand the user's high-level intentions so as to provide personalized interaction across sequences of applications/domains in two ways: 1) observe current context to predict appropriate follow-up domains; 2) understand the user's explicit and abstract cross-domain intentions. As a result, our model can 1) transition to the next app smoothly by predicting the follow-up domain given the current context and 2) assist the user at the task level by proposing a set of relevant apps given a task-level speech command. We also investigated methods to significantly reduce the gap between user-dependent models and user-independent models, since the latter may be less expensive in large scale. To convey the agent's understanding of the user's cross-domain intentions, we adopted key phrase extraction from user-generated language resources to refer to specific intentions. Our user study demonstrated that understandable references can be generated.

Based on what we have learned through this thesis, the following may be interesting extensions of the current work. In OOV learning, we want to investigate how to effectively expand the language understanding model to include the learned new words. For example, "Pittsburgh" in the user utterance "I'd like to try some *Pittsburgh* cuisine" is the new phrase learned by the agent (e.g., pronunciation and spelling are learned). We want the agent to automatically expand the existing [`cuisine_type`] class, rather than [`destination`], to incorporate "Pittsburgh". Sub-dialogs can be launched to ask the user for help if any ambiguity is presented. In cloud ASR adaptation, a fine-grained hypotheses combination model should be adopted to further improve the recognition accuracy. In user intention understanding, a system passively observing the user's behavior needs to analyze and then make judgements in the form of deciding to query the user about the intentions. Seeking help from the user can improve the quality of the machine-learned hypotheses, although causing human effort. Active learning should be deployed to optimize this learning process, balancing quality and effort. Given the current smart environment which hosts millions of applications/functionalities, it is time to re-think spoken dialog systems in this scale. Functionalities in such a large scale should be decoupled from the dialog manager and indeed become services to the intelligence. A bridge to commute between language (both the user's and the agent's) and appropriate services is a future direction in scaling up dialog systems [7, 10].

Bibliography

[1] Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. OpenFst: A general and efficient weighted finite-state transducer library. In Jan Holub and Jan Žďárek, editors, *Implementation and Application of Automata*, chapter 3, pages 11–23. Springer Berlin Heidelberg, 2007. 3.2

[2] James Allen, Nathanael Chambers, Lucian Galescu George Ferguson, Hyuckchul Jung, Mary Swift, and William Taysom. PLOW: A collaborative task learning agent. In *Proc. The National Conference on Artificial Intelligence*, volume 22, pages 1514–1519, 2007. 4.6

[3] Amos Azaria, Jayant Krishnamurthy, and Tom M. Mitchell. Instructable intelligent personal agent. In *Proc. The 30th AAAI Conference on Artificial Intelligence (AAAI)*, 2016. 4.6

[4] Issam Bazzi. *Modelling out-of-vocabulary words for robust speech recognition*. PhD thesis, Massachusetts Institute of Technology, 2002. 2.2.1

[5] Jerome R Bellegarda. Method and apparatus for a speech recognition system language model that integrates a finite state grammar probability and an N-gram probability. US Patent US6154722 A, 11 2000. 3.1

[6] Jerome R Bellegarda. Statistical language model adaptation: Review and perspectives. *Speech Communication*, 42(1):93–108, 2004. 3.1

[7] Jerome R Bellegarda. Spoken language understanding for natural interaction: The Siri experience. In Joseph Mariani, Sophie Rosset, Martine Garnier-Rizet, and Laurence Devillers, editors, *Natural Interaction with Robots, Knowbots and Smartphones*, chapter 1, pages 3–14. Springer New York, 2014. 5

[8] Michael W. Berry and Jacob Kogan. *Text mining: applications and theory*. Wiley, 2010. 4.4.1.3.2, 4.4.1.5

[9] Maximilian Bisani and Hermann Ney. Open vocabulary speech recognition with flat hybrid models. In *Proc. The 9th Biennial Conference of the International Speech Communication Association (Interspeech)*, pages 725–728, 2005. 2.2.1

[10] Nate Blaylock and James Allen. A collaborative problem-solving model of dialogue. In *Proc. The 6th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, 2005. 5

[11] Dan Bohus and Alexander I. Rudnicky. Constructing accurate beliefs in spoken dialog systems. In *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*,

pages 272–277, 2005. 4.4.1.3.2

[12] Dan Bohus and Alexander I. Rudnicky. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech and Language*, 23(3):332–361, 2009. 4.2

[13] Yun-Nung Chen and Alexander I. Rudnicky. Dynamically supporting unexplored domains in conversational interactions by enriching semantics with neural word embeddings. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pages 590–595. IEEE, 2014. 4.1, 4.4.1

[14] Yun-Nung Chen, Ming Sun, and Alexander I. Rudnicky. Leveraging behavioral patterns of mobile applications for personalized spoken language understanding. In *Proc. The 18th ACM International Conference on Multimodal Interaction (ICMI)*, pages 83–86, 2015. 4.1, 4.2, 4.4.1

[15] Yun-Nung Chen, William Yang Wang, Anatole Gershman, and Alexander I. Rudnicky. Matrix factorization with knowledge graph propagation for unsupervised spoken language understanding. In *Proc. The 53rd Annual Meeting of the Association for Computational Linguistics and The 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing (ACL-IJCNLP)*. ACL, 2015. 4.4.1

[16] Yun-Nung Chen, Ming Sun, Alexander I. Rudnicky, and Anatole Gershman. Unsupervised user intent modeling by feature-enriched matrix factorization. In *Proc. The 41st IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016. 4.4.1

[17] Grace Chung, Stephanie Seneff, Chao Wang, and Lee Hetherington. A dynamic vocabulary spoken dialogue interface. In *Proc. The 8th International Conference on Spoken Language Processing (ICSLP)*, pages 1457–1460, 2004. 2.3

[18] Dipanjan Das, Desai Chen, André FT Martins, Nathan Schneider, and Noah A Smith. Frame-semantic parsing. *Computational Linguistics*, 40(1):9–56, 2014. 2.3.4.1

[19] Richard Duda, Peter Hart, and David Stork. *Pattern Classification*. John Wiley and Sons, 2012. 4.4.1.4

[20] Jonathan G Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction (ROVER). In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 347–352, 1997. 4.4.1.4

[21] George W. Furnas, Thomas K. Landauer, Louis M. Gomez, and Susan T. Dumais. The vocabulary problem in human-system communication. *Communications of the ACM*, 30 (11):964–971, 1987. 2.3

[22] Kavita Ganesan, Chengxiang Zhai, and Jiawei Han. Opinosis: a graph-based approach to abstractive summarization of highly redundant opinions. In *Proc. The 23rd International Conference on Computational Linguistics (COLING)*, pages 340–348. ACL, 2010. 4.4.1.3.2

[23] Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. PPDB: The paraphrase database. In *Proc. The Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, pages 758–764, 2013. 2.3.2

[24] Helen Hastie, Marie-Aude Aufaure, Panos Alexopoulos, Hugues Bouchard, Catherine Breslin, Heriberto Cuayhuitl, Nina Dethlefs, Milica Gaic, James Henderson, Oliver Lemon, Xingkun Liu, Peter Mika, Nesrine Ben Mustapha, Tim Potter, Verena Rieser, Blaise Thomson, Pirros Tsiakoulis, Yves Vanrompay, Boris Villazon-Terrazas, Majid Yazdani, Steve Young, and Yanchao Yu. The Parlance mobile application for interactive search in english and mandarin. In *Proc. The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 260–262. ACL, 2014. 4.1, 4.2

[25] Hartwig Holzapfel, Daniel Neubig, and Alex Waibel. A dialogue approach to learning object descriptions and semantic categories. *Robotics and Autonomous Systems*, 56(11): 1004–1013, 2008. 2.3

[26] David Huggins-Daines, Mohit Kumar, Arthur Chan, Alan W. Black, Mosur Ravishankar, and Alexander I. Rudnicky. Pocketsphinx: A free, real-time continuous speech recognition system for hand-held devices. In *Proc. 31st International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 185–188, 2006. 2.3.4.1

[27] Anette Hulth. Improved automatic keyword extraction given more linguistic knowledge. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 216–223. ACL, 2003. 4.4.1.5

[28] Frederick Jelinek. Up from trigrams! - the struggle for improved language models. In *Proc. The 2nd European Conference on Speech Communication and Technology (Eurospeech)*, pages 1037–1040, 1991. 3.1

[29] Frederick Jelinek, Bernard Merialdo, Salim Roukos, and Martin Strauss. A dynamic language model for speech recognition. In *Proc. DARPA Workshop on Speech and Natural Language*, pages 293–295, 1991. 3.1

[30] Rosie Jones and Kristina Lisa Klinkner. Beyond the session timeout: automatic hierarchical segmentation of search topics in query logs. In *Proc. The 17th ACM conference on Information and knowledge management*, pages 699–708, 2008. 4.2

[31] Dietrich Klakow, Georg Rose, and Xavier Aubert. OOV-detection in large vocabulary system using automatically defined word-fragments as fillers. In *Proc. The 6th European Conference on Speech Communication and Technology (Eurospeech)*, 1999. 2.2.1

[32] Sylvia Knight, Genevieve Gorrell, Manny Rayner, David Milward, Rob Koeling, and Ian Lewin. Comparing grammar-based and robust approaches to speech understanding: A case study. In *Proc. The 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1779–1782, 2001. 3.1

[33] Quoc V. Le and Tomas Mikolov. Distributed representations of sentences and documents. In *Proc. The 31st International Conference on Machine Learning (ICML)*, pages 1188–1196, 2014. 4.4.1.4

[34] Christopher J. Leggetter and Philip C. Woodland. Maximum likelihood linear regression for speaker adaptation of continuous density hidden markov models. *Computer Speech and Language*, 9(2):171–185, 1995. 3.1

[35] Lee Hoi Leong, Shinsuke Kobayashi, Noboru Koshizuka, and Ken Sakamura. CASIS:

a context-aware speech interface system. In *Proc. The 10th international conference on Intelligent user interfaces (IUI)*, pages 231–238. ACM, 2005. 4.4.2.1

[36] Esther Levin, Shrikanth Narayanan, Roberto Pieraccini, Konstantin Biatov, Enrico Bocchieri, Giuseppe Di Fabbrizio, Wieland Eckert, Sungbok Lee, A. Pokrovsky, Mazin G. Rahim, P. Ruscitti, and Marilyn A. Walker. The AT&T-DARPA communicator mixed-initiative spoken dialog system. In *Proc. The 6th International Conference on Spoken Language Processing (Interspeech)*, pages 122–125, 2000. 4.2

[37] Qi Li, Gokhan Tür, Dilek Hakkani-Tür, Xiang Li, Tim Paek, Asela Gunawardana, and Chris Quirk. Distributed open-domain conversational understanding framework with domain independent extractors. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pages 566–571. IEEE, 2014. 4.2, 4.4.1

[38] Bor-shen Lin, Hsin-min Wang, and Lin-shan Lee. A distributed architecture for cooperative spoken dialogue agents with coherent dialogue state and history. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, volume 99, page 4, 1999. 4.1, 4.2, 4.4.1

[39] Hui Lin, Jeff Bilmes, Dimitra Vergyri, and Katrin Kirchhoff. OOV detection by joint word/phone lattice alignment. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 478–483, 2007. 2.2.1

[40] Pierre Lison. A hybrid approach to dialogue management based on probabilistic rules. *Computer Speech and Language*, 34(1):232–255, 2015. 4.2

[41] Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A. Smith. Toward abstractive summarization using semantic representations. In *Proc. The 14th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologie (NAACL-HLT)*, pages 1077–1086, 2015. 4.4.1.3.2

[42] Claudio Lucchese, Salvatore Orlando, Raffaele Perego, Fabrizio Silvestri, and Gabriele Tolomei. Identifying task-based sessions in search engine query logs. In *Proc. The 4th ACM International Conference on Web Search and Data Mining*, pages 277–286. ACM, 2011. 4.4.1.2, 4.4.1.4, 4.4.2.1

[43] Jean-Michel Lunati and Alexander I. Rudnicky. Spoken language interfaces: The OM system. In *Proc. Conference on Human Factors in Computing Systems*, pages 453–454, 1991. 4.1, 4.2, 4.4.1

[44] Olena Medelyan. *Human-competitive automatic topic indexing*. PhD thesis, University of Waikato, 2009. 4.4.1.3.2

[45] Marie Meteer and J. Robin Rohlicek. Statistical language modeling combining N-gram and context-free grammars. In *Proc. International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, volume 2, pages 37–40, 1993. 3.1

[46] Rada Mihalcea and Paul Tarau. TextRank: Bringing order into texts. In *Proc. Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 404–411. ACL, 2004. 4.4.1.5

[47] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. In *Proc. Workshop at International Conference on Learning*

Representations (ICLR), 2013. 2.3.2, 4.4.1.4

[48] George A Miller. WordNet: a lexical database for English. *Communications of the ACM*, 38(11):39–41, 1995. 2.3.2

[49] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted finite-state transducers in speech recognition. *Computer Speech and Language*, 16(1):69–88, 2002. 3.1

[50] Fabrizio Morbini, Kartik Audhkhasi, Kenji Sagae, Ron Artstein, Dogan Can, Panayiotis Georgiou, Shri Narayanan, Anton Leuski, and David Traum. Which ASR should I choose for my dialogue system? In *Proc. The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 394–403. ACL, 2013. 3.1

[51] Mikio Nakano, Shun Sato, Kazunori Komatani, Kyoko Matsuyama, Kotaro Funakoshi, and Hiroshi G Okuno. A two-stage domain selection framework for extensible multi-domain spoken dialogue systems. In *Proc. The 12th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 18–29. ACL, 2011. 4.1, 4.2, 4.4.1

[52] Aasish Pappu. *Knowledge Discovery through Spoken Dialog*. PhD thesis, Carnegie Mellon University, 2014. 2.2.1

[53] Aasish Pappu, Ming Sun, Seshadri Sridharan, and Alexander I Rudnicky. Situated multiparty interactions between humans and agents. In *Proc. The 15th International Conference on Human-Computer Interaction: interaction modalities and techniques*, pages 107–116, 2013. 4.1, 4.2

[54] Douglas B. Paul and Janet M. Baker. The design for the wall street journal based CSR corpus. In *Proc. Workshop on Speech and Natural Language*, pages 357–362, 1991. 2.2.3

[55] Fabian Pedregosa, Gaël. Varoquaux, Alexandre Gramfort, Vincent Michel, , Thirion Bertrand, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in Python. *The Journal of Machine Learning Research*, 12:2825–2830, 2011. 3.4

[56] Long Qin and Alexander I. Rudnicky. Finding recurrent out-of-vocabulary words. In *Proc. The 14th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2242–2246, 2013. 2.2.1

[57] Long Qin and Alexander I. Rudnicky. Learning better lexical properties for recurrent OOV words. In *Proc. Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 19–24, 2013. 2.2.1

[58] Long Qin and Alexander I. Rudnicky. Building a vocabulary self-learning speech recognition system. In *Proc. The 15th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 2862–2866, 2014. 2.2.1

[59] Long Qin, Ming Sun, and Alexander I. Rudnicky. OOV detection and recovery using hybrid models with different fragments. In *Proc. The 12th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 1913–1916, 2011. 2.2.1, 2.3.1, 2.3.4.2

[60] Long Qin, Ming Sun, and Alexander I. Rudnicky. System combination for out-of-

vocabulary word detection. In *Proc. The 37th International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4817–4820, 2012. 2.2.1, 2.3.1

[61] Antoine Raux, Brian Langner, Alan W. Black, and Maxine Eskenazi. LET'S GO: Improving spoken dialog systems for the elderly and non-native. In *Proc. The 8th European Conference on Speech Communication and Technology (Eurospeech)*, 2003. 4.1, 4.2

[62] Roni Rosenfield. Optimizing lexical and ngram coverage via judicious use of linguistic data. In *Proc. The 4th European Conference on Speech Communication and Technology (Eurospeech)*, pages 1763–1766, 1995. 1, 2.2, 2.3

[63] Alexander I Rudnicky, Jean-Michel Lunati, and Alexander M Franz. Spoken language recognition in an office management domain. In *Proc. International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 829–832. IEEE, 1991. 4.1, 4.2, 4.4.1

[64] Alexander I. Rudnicky, Aasish Pappu, Peng Li, Matthew Marge, and Benjamin Frisch. Instruction taking in the teamtalk system. In *AAAI Fall Symposium: Dialog with Robots*, pages 173–174, 2010. 4.6

[65] Seonghan Ryu, Jaiyoun Song, Sangjun Koo, Soonchoul Kwon, and Gary Geunbae Lee. Detecting multiple domains from users utterance in spoken dialog system. In *Proc. The 6th International Workshop on Spoken Dialogue Systems (IWSDS)*, pages 101–111, 2015. 4.2, 4.4.1

[66] Thomas Schaaf. Detection of OOV words using generalized word models and a semantic class language model. In *Proc. The 7th European Conference on Speech Communication and Technology (Eurospeech)*, pages 2581–2584, 2001. 2.2.1, 2.3.1

[67] Johan Schalkwyk, Doug Beeferman, Franoise Beaufays, Bill Byrne, Ciprian Chelba, Mike Cohen, Maryam Kamvar, and Brian Strope. Your word is my command: Google search by voice: A case study. In Amy Neustein, editor, *Advances in Speech Recognition*, chapter 1, pages 61–90. Springer US, 2010. 3.1

[68] Bill Schilit, Norman Adams, and Roy Want. Context-aware computing applications. In *Proc. The 1st Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 85–90, 1994. 4.4.2.1

[69] Xuehua Shen, Bin Tan, and ChengXiang Zhai. Implicit user modeling for personalized search. In *Proc. The 14th ACM International Conference on Information and Knowledge Management*, pages 824–831, 2005. 4.4.1.4, 4.4.2.1

[70] Choonsung Shin, Jin-Hyuk Hong, and Anind K. Dey. Understanding and prediction of mobile application usage for smart phones. In *Proc. ACM Conference on Ubiquitous Computing*, pages 173–182. ACM, 2012. 4.4.2, 4.4.2.3

[71] Craig Silverstein, Hannes Marais, Monika Henzinger, and Michael Moricz. Analysis of a very large web search engine query log. In *ACM Special Interest Group on Information Retrieval (SIGIR) Forum*, volume 33, pages 6–12, 1999. 4.2, 4.3.1

[72] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. BRAT: a web-based tool for NLP-assisted text annotation. In *Proc. The Demonstrations at the 13th Conference of the European Chapter of the Association for*

Computational Linguistics (EACL), pages 102–107. Association for Computational Linguistics, 2012. 4.3.2

[73] Andreas Stolcke. SRILM — an extensible language modeling toolkit. In *Proc. The 7th International Conference on Spoken Language Processing (ICSLP)*, pages 901–904, 2002. 3.2

[74] Hui Sun, Guoliang Zhang, Fang Zheng, and Mingxing Xu. Using word confidence measure for OOV words detection in a spontaneous spoken dialog system. In *Proc. The 8th European Conference on Speech Communication and Technology (Eurospeech)*, pages 2713–2716, 2003. 2.2.1

[75] Ming Sun, Alexander I. Rudnicky, and Ute Winter. User adaptation in automotive environments. In *Proc. The 4th International Conference on Advances in Human-Factors and Ergonomics (AHFE)*, pages 156–165, 2012. 4.1, 4.2

[76] Ming Sun, Yun-Nung Chen, and Alexander I. Rudnicky. Learning OOV through semantic relatedness in spoken dialog systems. In *Proc. The 16th Annual Conference of the International Speech Communication Association (Interspeech)*, pages 1453–1457, 2015. 4.4.1.4

[77] Ming Sun, Yun-Nung Chen, and Alexander. I. Rudnicky. Understanding user's cross-domain intentions in spoken dialog systems. In *NIPS workshop on Machine Learning for SLU and Interaction*, 2015. 4.1, 4.4.1

[78] Ming Sun, Yun-Nung Chen, and Alexander I. Rudnicky. HELPR a framework to break the barrier across domains in spoken dialog systems. In *Proc. The 7th International Workshop on Spoken Dialog Systems (IWSDS)*, 2016. 4.1, 4.4.1, 4.4.1.4

[79] Ming Sun, Yun-Nung Chen, and Alexander I. Rudnicky. An intelligent assistant for high-level task understanding. In *Proc. The ACM Conference on Intelligent User Interfaces (IUI)*, 2016. 4.1

[80] Robert Tibshirani, Guenther Walther, and Trevor Hastie. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 63:411–423, 2001. 4.4.1.4

[81] Johannes Twiefel, Timo Baumann, Stefan Heinrich, and Stefan Wermter. Improving domain-independent cloud-based speech recognition with domain-dependent phonetic post-processing. In *Proc. The 28th AAAI Conference on Artificial Intelligence (AAAI)*, pages 1529–1536, 2014. 3.1

[82] Keith Vertanen. Combining open vocabulary recognition and word confusion networks. In *Proc. The 33rd International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 4325–4328, 2008. 2.2.1

[83] Akos Vetek, John A. Flanagan, Ashley Colley, and Tuomas Keränen. Smartactions: Context-aware mobile phone shortcuts. In *Proc. The 13th Interactional Conference on Human-Computer Interaction (INTERACT)*, pages 796–799, 2009. 4.4.2

[84] Zhirong Wang, Tanja Schultz, and Alex Waibel. Comparison of acoustic model adaptation techniques on non-native speech. In *Proc. 2003 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 540–543, 2003. 3.1

[85] Wayne Ward and Sunil Issar. Recent improvements in the CMU spoken language understanding system. In *Proc. The workshop on Human Language Technology*, pages 213–216. ACL, 1994. 2.3.1

[86] Frank Wessel, Ralf Schlter, Klaus Macherey, and Hermann Ney. Confidence measures for large vocabulary continuous speech recognition. *IEEE Transactions on Speech and Audio Processing*, 9(3):288–298, 2001. 2.2.1

[87] Jason Williams, Antoine Raux, Deepak Ramachandran, and Alan Black. The dialog state tracking challenge. In *Proc. The 14th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 404–413. ACL, 2013. 4.6.2

[88] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: Practical automatic keyphrase extraction. In *Proc. The 4th ACM conference on Digital libraries*, pages 254–255, 1999. 4.4.1.3.2

[89] Steve Young. Using POMDPs for dialog management. In *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pages 8–13, 2006. 4.1, 4.2

[90] Ran Zhao, Alexandros Papangelis, and Justine Cassell. Towards a dyadic computational model of rapport management for human-virtual agent interaction. In *Proc. The 14th International Conference on Intelligent Virtual Agents*, pages 514–527, 2014. 4.4.1.2

[91] Victor W Zue and James R Glass. Conversational interfaces: Advances and challenges. *Proceedings of the IEEE*, 88(8):1166–1180, 2000. 2.3