**Rapid Transition to New Spoken Dialogue Domains: Language Model Training Using Knowledge from Previous Domain Applications and Web Text Resources**

*Murat Akbacak*, Yuqing Gao, Liang Gu, Hong-Kwang Jeff Ku

1Center for Spoken Language Research
University of Colorado at Boulder
Boulder, CO, USA 80303
murat@cslr.colorado.edu

2Human Language Technologies
IBM T. J. Watson Research Center
Yorktown Heights, NY, USA 10598
(yuqing,lianggu,hkuo)@us.ibm.com

**ABSTRACT**

In generic automatic speech recognition (ASR) systems, typically, language models (LMs) are trained to work within a broad range of input conditions. ASR systems used in domain-specific spoken dialogue systems (SDSs) are more constrained in terms of content and style. A mismatch in content and/or style between training and operating conditions results in performance degradation for the dialogue application. The main focus of this paper is to develop tools to facilitate rapid development of spoken dialogue applications within the context of language model training by focusing on the problem of automatically collecting text data that is useful to train accurate language models for the new target domain without manually collecting any in-domain data. We investigate a framework to extract useful information from previous domains and World Wide Web (WWW). We collect data by submitting queries to a search engine and then clean the resulting text via syntactic and semantic filtering. This is followed by artificial sentence generation. Without using any in-domain data, our system achieved a word error rate (WER) of 19.33%, a performance comparable to that achieved by a language model trained on manually collected 32K in-domain sentences. Using less than 1% of in-domain data along with the automatically generated text, our system achieved an ASR performance close to a language model trained on 60K in-domain sentences.

**1. INTRODUCTION**

Building a spoken dialogue system (SDS) is labor intensive since huge amounts of human transcribed speech are required to train statistical models employed within the ASR engine. The language model constitutes one of the statistical methods in ASR systems and the most commonly used approach is N-gram based language modeling which is easy to implement, but on the other hand, is sensitive to mismatches in content and style.

Because of the diversity of the language content among different dialogue domains, an LM trained for one domain does not work well for another domain because of content mismatch, even though the conversational style may be similar. When training data from domain independent applications are used, mismatch in both content and style occurs.

Early work has focused on the problem of rapid transition to new domains [1,2]. Some work [3,4,5,6] has been done to extract data from WWW to train language models for speech recognition applications. Previous studies [3,4,5] have shown how in-domain training data can be supplemented with text from the Web. In [3], conversational style text data is collected from web by submitting style-specific phrases (N-grams of words from Switchboard corpus) to capture conversational text, and topic-specific N-grams to capture content-specific text for a meeting transcription task. In a different study [6], web-based counts are used to improve language modeling. N-gram probabilities from web-based counts are interpolated with the unreliable trigram estimates of the corpus based language model. In [4], N-grams from the limited amount of in-domain data are used to search the web and other domains. Retrieved sentences are compared with each in-domain sentence, and filtered according to a similarity measure (BLUE).

All of the above methods for language model adaptation have the assumption that small amounts of LM training data are available in the target domain, and much larger amounts of text is available from other resources (other domains, WWW, etc.). Also, retrieved sentences from Web are used without any modification in spite of the fact that some part of the sentence improves the language model, but some other part introduces noise. In this study, we will present our solution to the problem of rapid transition to new dialogue domains within the context of language modeling. The novel part in our study is that no in-domain data is used to search the web, making it feasible to deploy a SDS without collecting real data. This scenario is useful when a demo system is desired to operate at a reasonable performance without any human effort. Initial text data consists of documents (emails, website, etc.) provided by the customer. We also investigate the effects of small amounts of in-domain data used in parallel with the filtered data.

In Section 2, different phases of SDS deployment are explained to see the motivation behind the problem. In Section 3, we explain how to extract text data using previous domain data and web text resources. In Section 4, we present our algorithm to filter and process data via syntactic and semantic filtering, and rule-based sentence generation. Section 5 talks about maximum entropy based sentence classification for LM clustering. Experimental results are given in Section 6. Conclusions and future work are presented in Section 7.

**2. SDS DEPLOYMENT**

Designing a domain specific SDS consists of three phases. The first phase involves deploying a demo system for the customer
by training acoustic and language models with the existing resources. This phase is important to convince the customer that the users will be able to interact with the system to complete a desired task.

In the second phase, Wizard-of-Oz (WoZ) data is collected from subjects and a pilot system is deployed. Usually, people are given a description of the dialogue application and some scenarios, based on which they are asked to write down what they would say.

After callers call in using the pilot system, speech data is collected and transcribed. These transcriptions are used to improve the accuracy of the acoustic model and language model of the dialogue system by supervised adaptation. Some of the previous studies [7] focused on making use of the collected speech data to improve the performance via unsupervised adaptation where collected in-domain speech data is automatically transcribed via ASR and then resulting transcripts are used to adapt the acoustic model and language model.

In this study, our main focus is to deploy a demo system automatically without manually collecting any in-domain data. To achieve this, we implemented the framework shown in Figure 1. Details of the framework will be explained in the following sections.

3. DATA COLLECTION

In our system, the goal is to extract the style information from previous domain data and content information from WWW. For a specific domain, we use a keyword vector to represent the content information of the domain as follows:

\[ K = [k_1, k_2, k_3, \ldots, k_N] \]

\( K \) is a vector of unigrams sorted according to their number of occurrences obtained from company documents. This vector can be modified manually if the dialogue designer thinks that there are other words that will also be frequently occurring during the application. We also have a vector \( G \) of words that are generic among different domains. This vector mainly consists of stop words and frequent domain independent words such as learn, know, get, account, information, one, Monday, dollar, etc.

\[ G = [g_1, g_2, g_3, \ldots, g_M] \]

We use previous domain data to extract generic chunks. A generic chunk is a sequence of generic words existing in vector \( G \). For example “I would like to get information on” is a generic chunk extracted from a previous domain data. During the construction of our search queries, we concatenate each generic chunk with domain specific keyword vector \( K \). The following is a sample search query:

“get information on” retirement plan ...

We can always refine our query by choosing shorter N-grams within the generic chunk if the number of hits returned is not sufficient. Queries are submitted to Google. In each retrieved document, HTML tags are stripped and sentences are extracted via a sentence boundary detector. Then we normalize the resulting text using an LDC Text Normalization tool. In addition to this data, we also borrow sentences from previous dialogue domains where above some percentage of the words in each sentence are generic. This threshold value is sentence-length dependent.

4. DATA ANALYSIS

For N-gram language modeling, there are two key issues playing important roles in the system performance. One of them is the quantity of training material since a language model should ideally have enough N-gram coverage. The other key issue is the quality of the language model since it should be concise. According to previous research, “effective language models can be built from very modestly sized corpora” [8]. Using all the data collected from the Web and other domains will yield high WERs. The solution that we use here is to analyze semantic and syntactic relations within the collected data, and then to generate artificial sentences using semantic filtering and rule-based syntactic transformations. By doing this, we do not use the whole sentence, but use only the relevant part as the training material.

4.1 Filtering Concepts

We use a syntactic parser to parse each sentence. Then, we run a noun phrase (NP) chunker to get all the NPs. We consider NPs as the concepts of the domain we are working on.

Using a seed NP-list, we decide if an NP is an in-domain NP or not. Here we will explain how we generate the seed list of NPs and use them to make the decision of in-domain NP vs. out-of-domain NP. We use internal company documents (such as email messages, company’s web page documents, etc.) to create the initial NP seed list. In our current system, context clues such as possessive adjectives/nouns (my, your, customer’s, etc.) are used to detect seed NPs. Among all the NPs extracted from the internal documents, we choose the ones containing a possessive adjective or noun.

After creating the seed NP list, we train a bigram language model from this list. We call this the seed-NP model. The seed NP list also contains generic NPs as well. During seed-NP model training, we consider stop words as out-of-vocabulary (OoV) words and we do not consider them during bigram probability calculation.

Figure 1. System diagram
For the classification of an unseen NP (in-domain vs. out-of-domain), we can use two methods. The first method is to calculate the perplexity of the word sequence of the NP using the seed-NP model we trained from the seed-NP list, and then generate a distribution. Based on this perplexity distribution, we set a threshold. If the perplexity of an NP in a WWW-sentence is above this threshold, we make the decision that it is an in-domain NP. The second solution to the classification problem can be to find the most frequent N-grams in the seed-NP list and classify an unseen NP as an in-domain NP if there is an exact bigram match. An example seed NP list is shown in Table 1.

<table>
<thead>
<tr>
<th>Example Seed NP List</th>
</tr>
</thead>
<tbody>
<tr>
<td>my 401k plan</td>
</tr>
<tr>
<td>your allowable contrs</td>
</tr>
<tr>
<td>my annual retirement</td>
</tr>
<tr>
<td>my retirement fund</td>
</tr>
<tr>
<td>my annual statement</td>
</tr>
<tr>
<td>customer’s account balance</td>
</tr>
<tr>
<td>your password</td>
</tr>
</tbody>
</table>

Table 1: An example seed NP list.

### 4.2 Filtering Actions

In a dialogue scenario, the user may wish to perform an action (transfer a balance, cancel transaction, etc.). In this context, each verb corresponds to an action. As we classify concepts into in-domain and out-of-domain concepts, we do the same thing for the actions as well. We again use context clues to generate a seed action list. Context clues used here are conversational phrases such as “I want/would like to VB”, “if you VB”, “you may VB”, etc. All the actions attached to these conversational phrases are listed as initial seed actions. Then, the initial seed action list is pruned by removing the ones that are not attached to an in-domain NP. For those actions that are not in the initial seed list, we created search queries such as “I want to VB” k1 k2 ... k3 and submit them to Google. The retrieved sentences are parsed to see if there is any occurrence of this action where it is attached to an in-domain NP.

### 4.3 Artificial Sentence Generation

We created syntactic templates using previous domain data to artificially generate in-domain sentences using in-domain concepts and the actions detected above. For example, if we have a sentence collected from the Web that contains in-domain concept and action, we look at the syntactic parse of that sentence. If it appears in one of the templates, we transform the sentence to conversational style (example is shown in Table 2).

```
[These charges can be paid through the telephone]  
[These charges]_NP [can]_MD [be paid]_VBN ...

Transformation rule: [NP MD be VBN] -> I’d like to [VB NP]
```

Table 2: Rule based artificial sentence generation.

Current transformation rules are written manually, but it is possible to extract these rules automatically by examining a domain that has enough in-domain data. We also use previous domain data to inherit conversational sentence structure via NP replacement.

### 5. DATA CLUSTERING

After generating sentences, we can use them directly to train a language model for the ASR system. In the case that the dialogue state information is provided during run-time, we should be able to benefit from this prior knowledge. This requires clustering the artificially generated sentences into different dialogue states. Here, we consider the case where we have very limited amount of in-domain WoZ data with dialogue state information, and we can train a classifier and assign dialogue states to the sentences as in a classification problem.

#### 5.1 Maximum Entropy Based Classifier

We first remove generic sentences from the limited amount of in-domain WoZ data. After tagging numbers, dates, amounts, etc., we train a Maximum Entropy (Maxent) classifier on a limited amount of training data. The Maxent classifier [9] is a flexible modeling framework that enables the combination of multiple features that are not necessarily statistically independent. The features are combined as follows:

\[
P(S | W) = \frac{\exp(\sum_\lambda \lambda_i f_i(S, W))}{\sum_\lambda \exp(\sum_\lambda \lambda_i f_i(S', W))}
\]

This equation describes the posterior probability of a particular dialogue state S, given the word sequence W spoken by the caller. Note that the denominator includes a sum over all states S’, which is a normalization factor for probabilities to sum to 1. The \( \lambda_i \) are indicator functions, or features, which are “activated” based on computable features on the word sequence. For example, we use a combination of unigram (single word) and bigram (word pair) features in this paper. Since the classifier is not well trained because we only have a limited amount of training data, we consider N-best dialogue states for each sentence rather than just assigning the best dialogue state. Another way might be to take classifier posterior probabilities into consideration during the N-gram probability calculation.

#### 6. EXPERIMENTS

##### 6.1 Application

We perform our experiments on an application where users perform financial transactions on their retirement accounts. We have 75K sentences collected in a WoZ setup. The baseline language model is a deleted interpolation trigram model. It was built using a vocabulary of 3228 words. In all cases, 90% of the data is used for training and 10% is used as a held-out set for smoothing N-gram probabilities. The dialogue state specificity was introduced by using the dialogue state as the sentence start context for the model. Then at system run time the dialogue state was used to prime the language model. We generated results for the following cases: (a) without-dialogue-state and (b) with-dialogue-state. The test data consists of 3148 utterances. Acoustic models are trained on generic telephony data.
6.2 Recognition Results

Using language models trained on domain independent dictation or large vocabulary speech recognition tasks resulted in fairly high word error rates (>45%). A language model trained on data collected from WWW (not cleaned) yielded a word error rate higher than 30%. We also trained language models using previous application specific (but different domain) data, and the lowest WER was 29.11% when the data from a bank transaction system is used for LM training. Figure-2 shows WER results for different amounts of in-domain data. As expected, WERs are lower when the dialogue state information is provided.

![Figure 2. WER results for each training data size (a) without and (b) with dialogue state information.](Image)

In our experiments, we first used collected and cleaned data without using any in-domain data. 150K sentences are collected from WWW and previous domains as explained in Section 3. A WER of 26.20% was obtained by using this data to train the LM. After filtering and transforming this data with the methods presented in Section 4, we achieved a WER of 19.33%. This WER is close to that of a language model trained on 32K in-domain sentences without using dialogue state information. At this point, we randomly chose 500 sentences from the in-domain data, and added this to the artificially generated data. The result was interesting since the WER dropped to 18.13%-18.40%. This happens not because unseen trigrams are introduced by adding in-domain data, but because the probability distribution of the existing trigrams has changed.

Another way of changing the probability distribution is clustering the sentences. This corresponds to having a dialogue state attached to each artificially generated sentence. In this dialogue system, there are 32 dialogue states, 11 of which are related to some type of confirmation. We randomly chose 40 non-generic sentences from the in-domain data for each dialogue state and trained a bigram based maximum entropy classifier. The accuracy of this classifier is 75% on the training set due to the limited amount of training material. For each non-generic sentence in the final set (filtered and transformed), we assigned the top 5 dialogue states. The reason for assigning 5 dialogue states rather than 1 dialogue state to each sentence is that our limited amount of training data resulted in a classifier that is not very accurate. Using the artificially generated data set with dialogue state assignments, a WER of 18.47% is obtained. Interpolating this LM with the one trained from 2K in-domain sentences with dialogue state information gives a WER of 15.43%. Assigning dialogue states to the artificially generated sentences, by using a very limited amount of training data for the classifier, improves the recognition accuracy further more. This shows that even a very limited amount of in-domain data is useful to capture dialogue state information.

7. CONCLUSIONS AND FUTURE WORK

In this study, we presented a new framework to build a language model for a new spoken dialogue domain without specifically collecting conversational speech/language data for that domain. Our framework employs conversational patterns learned from previous spoken dialogue applications, and domain specific concepts from the company documents (e.g., online website), to generate queries to retrieve sentences from Web. Retrieved sentences are filtered and transformed via syntactic and semantic filtering, and rule based sentence generation, respectively. The current system is tested in a dialogue application where users perform transactions on their retirement accounts. Without using any in-domain data, our system achieved a WER of 19.33%, a performance that is achieved by a language model trained on 32K of manually collected in-domain sentences. Using less than 1% of in-domain data along with the automatically generated text, our system achieved a performance close to that achieved with 60K in-domain sentences. Results are very promising, and suggest a viable procedure to follow for future advances. Future work will focus on testing the new framework in other domains, as well as using different amounts of in-domain data during data collection, analysis, and clustering.

8. REFERENCES


