IMPACT OF WEB BASED LANGUAGE MODELING ON SPEECH UNDERSTANDING

Ruhi Sarikaya, Hong-Kwang Jeff Kuo and Yuqing Gao
IBM T.J. Watson Research Center
Yorktown Heights, NY 10598
{sarikaya,hkuo,yuqing}@us.ibm.com

ABSTRACT

Data sparseness in building statistical language models for spoken dialog systems is a critical problem. In a previous paper we addressed this issue by exploiting the World Wide Web (WWW) and other external data sources in a financial transaction domain. In this paper, we evaluate the impact of improved speech recognition due to Web-based Language model (WebLM) on the speech understanding performance in a new domain. As speech understanding system we use a natural language call–routing system. Experimental results show that the WebLM improves the speech recognition performance by 1.7% to 2.7% across varying amounts of in–domain data. The improvements in action classification (AC) performance were modest yet consistent ranging from 0.3% to 0.8%.

1. INTRODUCTION

In recent years spoken dialog systems (SDS) matured in such a way that it has been widely used in many applications. Nevertheless, rapid deployment of SDS for new domains remains as a challenge. One of the main problems facing us today is the data sparseness for building robust statistical language models. Acoustic modeling is fairly insensitive to the domain of application as long as there is not significant environmental and channel mismatch between training and test conditions. Language modeling on the other hand, has significant impact on the speech recognition performance. Using LMs built for fairly generic applications such as dictation or Switchboard [12] does not help to improve system performance to a usable level. In such circumstances it is essential to improve the LM performance. The bulk of the LM research has concentrated on improving the language model probability estimation, while obtaining additional training material from external resources received little attention [2, 1].

In [8], we took advantage of the WWW and available external text resources previously collected for other applications to improve LM performance. In fact, as the largest data source, WWW has been previously utilized for numerous natural language processing (NLP) applications [3]. The Web has been used for language modeling as well [1, 4, 5]. The contribution of our study was to propose a framework on how to use the Web for limited domain spoken dialog systems. To this end, we formulated a query generation for data retrieval from the Web and developed a data filtering and selection mechanism to extract the useful utterances from the retrieved pages [8]. The proposed mechanism makes good use of limited in–domain data to sift through the large external text inventory to identify “similar” sentences. In many, if not all, of the previous work, documents were used as the unit in accepting or rejecting training material. We believe that going one step further and sifting for relevant information within a document is essential. Therefore, we do not take the returned documents as a whole, but rather find relevant utterances in the page. By doing so, we filter out irrelevant text and keep only relevant data for language modeling.

In reality when we start to build a SDS for a new domain, the amount of in–domain data for the target domain is usually small. In cases where there is no in–domain data, we generate artificial data. Using available data, we build a pilot system that is mainly used to collect real in–domain conversational data. It is essential to have a pilot system that operates with a reasonable accuracy both at the speech recognition and speech understanding level so that the users are not frustrated in communicating with the SDS.

In this paper, we apply the previously proposed mechanism for Web–based language modeling to a new domain which involves a large company’s call center customer hotline for technical assistance. Furthermore, we evaluate the impact of the speech recognition performance improvement on the speech understanding performance. Here, as speech understanding application we use a natural language call–routing system [9].

The rest of the paper is organized as follows. In Section 2 we present the framework to collect relevant data from the Web as well as other available resources for language modeling. In Section 3, we explain how search queries are generated and sentences are selected from the retrieved pages. A brief description of the natural language call–routing application is given in Section 4. Section 5 presents the experimental results. Section 6 summarizes the findings.

2. THE MECHANISM FOR WEB DATA COLLECTION

The proposed framework utilizes not only the Web, which we refer as a dynamic resource, but also other static resources. Static resources include any corpus that is previously collected for limited–domain or domain–independent applications. We gave a list of the static corpora used in our experiments in [8].

The flow–chart for the proposed approach is depicted in Fig. 1. We assume that we are given some limited data belonging to the target domain. This data can also be generated manually. We generate queries from these sentences and search the Web. The retrieved documents are filtered and the documents are further processed to extract relevant utterances using the limited in–domain data. We employ a similarity metric to identify
sentences that are likely to belong to the target domain. The same process is applied to the static data sources. However, for the static corpora, in–domain data is directly used with the similarity metric to identify relevant utterances without query generation and retrieval. Finally, we build a domain–specific language model using the relevant sentences obtained from static and dynamic sources. An effective method to combine a small amount of in–domain and a large amount of out–of–domain data is through building separate language models and interpolating them [7].

3. SEARCH QUERY GENERATION AND SENTENCE SELECTION

A set of initial experiments with several search engines made us believe that Google was the most useful for our application. Google indexes web pages (it also includes URLs that it has not fully indexed) and many additional file types in the web database. We did not want to increase computation by the conversion of non–text documents into plain text. We only downloaded those files where text can be retrieved efficiently.

Search query generation from a sentence is a key issue. The queries should be sufficiently specific since the more specific the query is, the more relevant the retrieved pages are. On the other hand, if the query is too specific there may not be enough, if any, retrieved pages. In reality, we do not have infinite resources as such, one needs to avoid sending too many failed requests to the server just to get documents for a sentence. Therefore, the approach we take for query generation takes these concerns into account by generating queries that start from the most relevant case and gracefully degrade to the least relevant case. We define the most relevant query as the one that has ANDed maximal n–grams with context. The least relevant query is defined as the ORed unigrams obtained from an utterance.

An example of query generation is given in Table 1. The first step in forming queries is to define a set of frequently occurring words as stop words (i.e. is, the, a, as, etc.). The remaining text is chunked into n–gram islands consisting of only content words. Then, we add context to these islands by including their left and right neighbors. The purpose of adding context around the content words is to incorporate conversational style into queries

| (S) | what is the balance of my stock fund portfolio |
| (Q1) | [what] [balance] [stock fund portfolio] |
| (Q2) | [what] [balance] [stock fund portfolio] |
| (QN) | [what] [balance] [stock fund portfolio] |

Table 1: Query generation.
data as well as static corpora where the similarity score is above an empirically determined threshold, which is determined based on word error rate using a held-out data and is set to 0.08.

4. NATURAL LANGUAGE CALL–ROUTING

The aim of call–routing is to understand the speaker’s request and take the appropriate action. Typically, natural language call–routing requires two statistical models. The first performs speech recognition that transcribes the spoken utterance. The second is the Action Classifier (AC) model that takes the spoken utterance obtained by the speech recognizer and predicts the correct action to fulfill speaker’s request. Actions are the categories into which each request a caller makes can be mapped.

In this work, we use the MaxEnt method to build a statistical classifier [9]. The MaxEnt method is a flexible modeling framework that allows the combination of multiple overlapping information sources [10, 11]. The MaxEnt modeling matches the feature expectations exactly while making as few assumptions as possible in the model. The multiple information sources are combined as follows:

\[
P(C|W) = \frac{\sum_{C} \lambda_i f_i(C, W)}{\sum_{C'} \sum_{i} \lambda_i f_i(C', W)},
\]

which describes the probability of a particular class \(C\) (e.g. action class) given the word sequence \(W\) spoken by the caller. Notice that the denominator includes a sum over all classes \(C'\), which is essentially a normalization factor for probabilities to sum to 1. The \(f_i\) are indicator functions, or features, which are “activated” based on computable features on the word sequence, for example if a particular word or word pair appears, or if the parse tree contain a particular tag, etc. For simplicity, we only use unigram (single word) features in this paper, also commonly known as a “bag of words” model. The MaxEnt models are trained using the improved iterative scaling algorithm [10] with Gaussian prior smoothing [11] using a single universal variance parameter of 2.0.

5. EXPERIMENTAL RESULTS AND DISCUSSION

The domain of call–routing application in this paper is a technical support hotline for a Fortune–500 company’s call center [9]. There are 35 predetermined call–types. The full training data has 27K utterances amounting to 177K words. This data is split into ten chunks by uniformly sampling from the full set. The first chunk is further split into two 1.3K chunks. The vocabulary has 3667 words. The acoustic models are first trained using +1000 hours of generic telephony acoustic data, later MAP adapted using in–domain training data to this application. A decision tree clustered context dependent modeling with continuous density HMMs is used to model acoustic space. There are 2198 context dependent states with 222K Gaussians in the acoustic model. A separate test data consisting of 5644 utterances is used for evaluation.

The trigram language models are built with deleted interpolation. In all cases the data is split into a 90% and 10% chunks. The former chunk is used for training and the latter chunk is used as heldout set for smoothing. Using a language model built for domain independent dictation or large vocabulary speech recognition tasks resulted in fairly high error rates (> 45%). In Fig. 2, we plotted the word error rates (WER) for five language models with respect to the amount of in–domain data. In the figure, “BASE–LM” stands for the baseline LM built using only the in–domain data, “WEB–LM” stands for the LM built using only the WEB data and “STAT–LM” is for the LM built using only the data obtained from static resources. We also presented interpolations of these LMs. The very first points on the curves correspond to using only 1.3K in–domain sentences. The WER for the BASE–LM is 30.9% and the corresponding figure for the WEB–LM is 35.1%. Interpolating WEB–LM for the STAT–LM reduces the WER to 32.5%. Interpolation of WEB–LM with the BASE–LM results in 28.9%. Three way interpolation of BASE–LM, WEB–LM and STAT–LM reduces the figure to 28.2%. This is 2.7% reduction compared to BASE–LM WER. Similar improvements are observed up to 8K in–domain data. As we include more in–domain data and all the WEB data collected for each chunk are treated as the static data for the next chunk, the amount of data to be processed easily reached to more than 10GB. Static resources account for much of the computation (about 80% for this application). We observe that at 8.1K in–domain data using static resources in addition to Web data contributes marginally to improved performance.

An alternative way to look at Fig. 2 is to see the needed in–domain data reduction to achieve a given performance level. We observe about 3-to-4 fold reduction to match the in–domain LM performance. For example, we can match the performance of in–domain LM that uses full 27K set by using 8.1K in–domain LM, the Web and Static LMs.

We did not plot the graph for WEB–LM, STAT–LM and their interpolations with the BASE–LM after 8.1K in–domain utterances. Nevertheless, we wanted to see the impact of interpolating WEB–LM and STAT–LM with the full 27K BASE–LM. In Fig. 3, we plotted the WER against the amount of in–domain data used to retrieve data for the WEB–LM and STAT–LM and their interpolation with the full 27K in–domain LM (27K–BASE–LM). Note that the first points (for 1.3K) for WEB–LM and “WEB–LM+STAT–LM” are same as those in Fig. 2. When these LMs are interpolated with the 27K–BASE–LM, the WER
Lastly, in Table 2, we presented the AC accuracy figures corresponding to the lower two curves at the bottom of Fig. 3. The improvements in speech recognition do not translate to improvements of same magnitude in AC accuracy. However, there is again some small improvement as given in the table.

### 6. CONCLUSIONS

We considered the WWW and the available external static corpora as possible sources for building robust statistical language models. We proposed a framework to retrieve, filter, and select utterances from these sources by diligently exploiting the limited amount of available in-domain data. The amount of in-domain data required to achieve a given performance level is reduced by a factor of 3 to 4. The improvements in speech recognition accuracy range from 1.7% to 2.7%. The resulting improvements in call-routing range from 0.3% to 0.8%.

### References


<table>
<thead>
<tr>
<th>LM</th>
<th>AC Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>27K-BASE-LM</td>
<td>83.3</td>
</tr>
<tr>
<td>27K-BASE-LM + 1.3K [WEB-LM+STAT-LM]</td>
<td>83.7</td>
</tr>
<tr>
<td>27K-BASE-LM + 2.7K [WEB-LM+STAT-LM]</td>
<td>83.6</td>
</tr>
<tr>
<td>27K-BASE-LM + 5.1K [WEB-LM+STAT-LM]</td>
<td>84.0</td>
</tr>
<tr>
<td>27K-BASE-LM + 8.1K [WEB-LM+STAT-LM]</td>
<td>83.8</td>
</tr>
</tbody>
</table>

Table 2: AC Performance (%) for Various Language Models.