# BUILDING TOPIC SPECIFIC LANGUAGE MODELS FROM WEBDATA USING COMPETITIVE MODELS

*Abhinav Sethy, Panayiotis G. Georgiou, Shrikanth Narayanan*

Speech Analysis and Interpretation Lab
Department of Electrical Engineering-Systems
Viterbi School of Engineering
University of Southern California

## ABSTRACT

The ability to build topic specific language models, rapidly and with minimal human effort, is a critical need for fast deployment and portability of ASR across different domains. The World Wide Web (WWW) promises to be an excellent textual data resource for creating topic specific language models. In this paper we describe an iterative web crawling approach which uses a competitive set of adaptive models comprised of a generic topic independent background language model, a noise model representing spurious text encountered in web based data (Webdata), and a topic specific model to generate query strings using a relative entropy based approach for WWW search engines and to weight the downloaded Webdata appropriately for building topic specific language models. We demonstrate how this system can be used to rapidly build language models for a specific domain given just an initial set of example utterances and how it can address the various issues attached with Webdata. In our experiments we were able to achieve a 20% reduction in perplexity for our target medical domain. The gains in perplexity translated to a 4% improvement in ASR word error rate (absolute) corresponding to a relative gain of 14%.

## 1. INTRODUCTION

A key step in creating speech recognition systems for different domains and applications is to identify the right text resources for building language models matched to that application or domain. In some cases text for the target domain might be available from institutions such as LDC and NIST. However in most cases such data are not readily available and needs to be collected manually. This imposes severe constraints in terms of both the system turnaround time and cost. To limit the effects of data sparsity, a topic independent language model is often merged with a language model generated from limited in-domain data to generate a smoothed topic specific language model. However this adaptation approach can only be viewed as a procedure to reduce the effect of data sparsity and will likely give suboptimal results compared to having good in-domain data.

The World Wide Web (WWW) promises to be a good resource to automatically gather data for building domain specific language models[1][2]. The amount of text available on the WWW is immense (more than 4 Billion pages indexed by Google alone) but it is difficult to access clean text relevant to a particular domain directly. To gather appropriate data, carefully crafted queries should be generated for the specific domain. However, even with the best queries, a large section of the returned documents, or parts of them, might not actually be relevant for the particular task and tend to be noisy. Thus it becomes important to selectively weight the downloaded Webdata so as to maximize the gains in terms of language model perplexity or ASR Word Error Rate (WER) reduction.

In this paper, we address the case where an initial representative set of documents for the domain of interest is available. If this seed set is large, then the web data acquisition can be seen as an update to the already existing language model. This is of special interest in cases where the speech recognition needs to handle new content such as in broadcast news applications[3]. However if the initial set of documents is too small to build a robust language model, then the web data plays a more important role. In addition to the initial topic model, we assume the existence of a generic topic independent language model and the corresponding documents on which it was built. The two language models, one topic dependent and the other topic independent (hereby referred to as the background model) are used to generate search queries using a relative entropy measure described in [4] and weight the downloaded data at the utterance level in a soft clustering fashion. A rejection model is built from low scoring downloaded web data to help reject spurious text and documents like inline advertisements, cross links etc in the next iteration of web downloads. To supplement the utterance level processing we use document classification techniques based on TFIDF and Naive Bayes to generate query words and provide document level weights.

Our approach differs from previous work on use of web data for ASR in two aspects The relative entropy based query generation mechanism enables us to download documents which have a higher match with the in-domain topic language model than using count based measures[5]. Instead of doing data selection at the utterance[6] or document level[7] we weight the downloaded webdata for its relevance to the domain of interest using both document and utterance match criteria. In addition we track the perplexity gains we observe from specific query terms and URLs to refine our search for the next iteration of downloads.

The paper is organized as follows. In the next section we describe the process of generating keywords and keyphrases using relative entropy (R.E) between language models. In Section 3 we will describe our utterance scoring and clustering mechanism. Section 4 describes the document classification techniques we used and how they are combined with the rest of the system. The experimental setup and results are discussed in Section 5. We conclude with an analysis of our results and directions for future work.

## 2. GENERATING QUERIES USING RELATIVE ENTROPY

To generate query strings we first compare the topic language model with the background language model using the relative entropy measure first described in [4]. In general, relative entropy computation between two discrete distributions requires density comparisons across all the possible symbols in the alphabet. This implies that a direct R.E implementation for n-gram language models would require $V^n$ computations, where $V$ is the vocabulary size. This would make R.E comparisons for even medium sized trigram LM's with 15-20K words computationally prohibitive. However since real world n-gram language models are tree structured and a majority of the n-gram densities backoff to probabilities of corresponding $n-1$ grams, it becomes possible to compute R.E between two LM's in $O(L)$ computations where $L$ is the number of language model terms actually present in the two LM's. The computation process described in [4] recursively calculates the relative entropy for an n-gram model using the relative entropy for the n-1 gram model. During the computation we are able to get relative entropy conditioned on word sequence histories $h$ , i.e the R.E between the n-grams represented by $p(x/h)$ and $q(x/h)$ where $h$ is the history on which the probability of seeing the word $x$ is conditioned; $p$ being the base (topic model) LM and $q$ being the LM (background model) being evaluated with respect to $p$.

The histories with large R.E serve as good candidates for being keyphrases or keywords since they have good discriminative power. To make sure that they qualify as key words or phrases for the topic, we also need to ensure that $p(h)$ is larger than the corresponding $q(h)$. For the medical domain task which we describe in our experiments some of the keyphrases were very relevant ("Stomach Ache", "Feeling Nausea", "Bad Headache"). However in many cases key phrases contained words without key nouns such as "Hurts The", "Place A". We found out that even though these query phrases are not very useful on their own, they are effective when combined with keywords, e.g. "Hurts The "+"Stomach". A detailed discussion on the effect of query length and information retrieval accuracy can be found in [8].

A list of query keyphrases and keywords was generated using the techniques described above and merged with a keyword list based on the mutual information between words and class labels using the document classification system described in Section 4. Then a random selection of five query words with a couple of randomly chosen keyphrases was used as a search query (See Section 5)for the Google SOAP API which then returned the relevant set of URLs. The mix of number of keywords and keyphrases to embed in a single query is a function of the task at hand. For example, to model conversational styles it should be advantageous to put more keyphrases in the query than keywords[7]. The set of URLs was then downloaded and converted to text, which was subsequently weighted and filtered as described in the next section.

## 3. WEBDATA SELECTION AND WEIGHTING

Likelihood scores for every utterance were calculated using the background, topic and rejection language models. Utterances that scored high with the rejection model or scored low with background and topic model with respect to the average utterance score were removed from the update process for the background and topic model. For other utterances relevance weights were calculated using the expression

$$\text{score}(T) = \frac{P(\text{utt}/T) * DW(T)}{P(\text{utt}/T) * DW(T) + P(\text{utt}/B) * DW(B)} \quad (1)$$

$$\text{score}(B) = \frac{P(\text{utt}/B) * DW(B)}{P(\text{utt}/T) * DW(T) + P(\text{utt}/B) * DW(B)} \quad (2)$$

$T$: Topic Model
$B$: Background Model
$DW(T)$: Document weight for topic
$DW(B)$: Document weight for background

The document weight was calculated from the document level models described in section 4. The document level weights were included as a trade-off between the relevance of the entire document and that of the given utterance. Utterances weighted in this fashion were grouped into a fixed number of bins by their weight for the topic model and the background model. This binned data was then used to create language models which were combined according to their assigned weights to create the update topic and background models. The update models were subsequently merged with the initial topic and background model with the merge weight determined by iterative perplexity minimization on a held-out set[9].

We observed that the downloaded web data contained a lot of spurious text corresponding to content like advertisements, links and embedded subsections from other content (common in news and reviews pages). These items typically add a lot of additional terms in the vocabulary and also have a detrimental effect on the language model performance (See Section 5). To remove these text items we included a rejection model in our data selection and filtering phase. To initialize the rejection model in the first iteration of downloads, documents whose scores were very low compared to the average document scores for the background and topic model were included in the data for the rejection model and a LM was built on this data. Utterances were also filtered using the same strategy. In subsequent iterations, documents classified as being close to rejected documents and utterances with high likelihood match to the rejection language model were included in the rejection model.

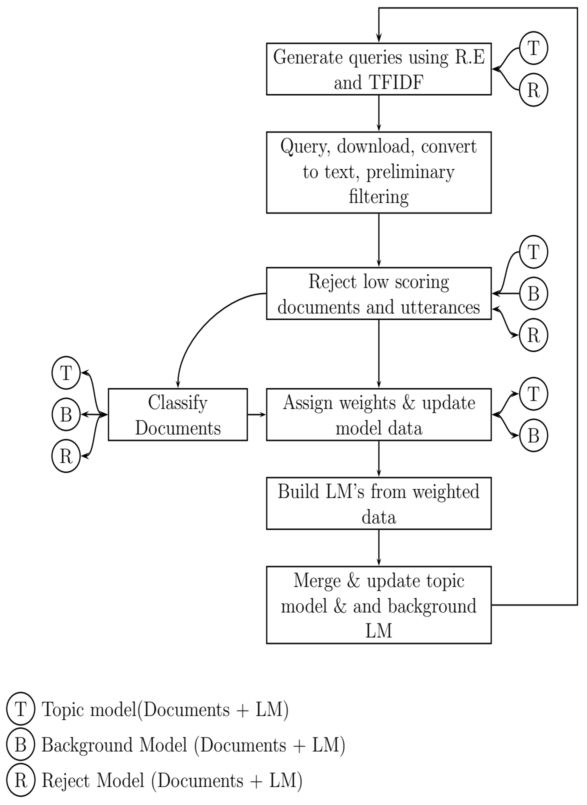
The queries and associated URLs are evaluated in terms of the average perplexity of the downloaded text with the topic and the background models. In the next iteration, we prefer queries that gave the best match with the topic model. Queries and URLs which correspond to documents that score poorly in perplexity terms or are rejected by the rejection model are not used in subsequent iterations.

## 4. DOCUMENT LEVEL PROCESSING

The documents corresponding to the topic, background and rejected data were used to train a document classification system based on the TFIDF/Naive Bayes measure using the CMU BOW toolkit[10]. The document classification system was used to assign document weights to the downloaded text for the three classes: background, topic and rejected data. These weights are used in conjunction with the utterance level weights as described in the previous section.

The training set is enhanced with each iteration, as downloaded documents are included. Before including a particular downloaded document into the training set of either the background model or the topic model, we first filter out the rejection model utterances and utterances with low relevance weight for the model.

Figure 1 presents an overview of the entire system. The termination condition for the iteration loop can be set in terms of a number of parameters like size of the downloaded data, perplexity improvements, keyword coverage etc.



Generate queries using R.E and TFIDF
Query, download, convert to text, preliminary filtering
Reject low scoring documents and utterances
Classify Documents
Assign weights & update model data
Build LM's from weighted data
Merge & update topic model & and background LM
T
B
R
T Topic model(Documents + LM)
B Background Model (Documents + LM)
R Reject Model (Documents + LM)

**Fig. 1**. System overview. For the first iteration the system has a dummy reject model.

## 5. EXPERIMENTAL SETUP AND RESULTS

Our experiments were conducted on medical domain data as part of the Transonics speech to speech translation system[11]. 3K utterances from patient-doctor dialogs were used to build the topic model. 1K utterances were used for testing and about 500 utterances were used as heldout data for tuning system parameters such as language model weights. For the background model we chose to use a interpolated model consisting of data from SWB (4M words), WSJ (3M words) and some text from the Gutenberg project (2M words). The Gutenburg project is a collection of ebooks of English literature. After pruning[12] the background model had a vocabulary of about 40K words The test set size was fixed at 110K words. The downloaded data size was kept fixed at 10M words to provide meaningful comparisons across different experiments.

| Initial data size (K words) | PPL with initial data | PPL with initial data+Webdata |
|---|---|---|
| 5 | 200 | 120 |
| 10 | 160 | 110 |
| 15 | 117 | 92 |

**Table 1**. Perplexity(PPL) of testdata for different sizes of initial seed data. Both the models were merged with the baselm using linear interpolation

| Keyword count | Keyphrase count | Perplexity of merged LM |
|---|---|---|
| 5 | 2 | 92 |
| 4 | 1 | 100 |
| 3 | 0 | 111 |

**Table 2**. Effect of different number of query keywords and keyphrases on system performance.

### 5.1. Effect of initial seed set size

In our first experiment we evaluated the performance of the system for different sizes of the initial seed set. The goal was to see how important it is to have a large initial set for the query generation subsystem to generate a good set of keywords and keyphrases. The results can be seen in Table 1. The results indicate that the query generation process performs nicely even with small amounts of data and hence the performance of the final merged models is not critically dependent on the size of the seed set. With decreasing seed set size, the jump in perplexity for the models trained from the seed data is very high as compared to the final models built after merging with Webdata. The background model was also used as the base language model for interpolation with initial data.

### 5.2. Effect of query size and type

We studied the effect of different query structures on the system performance. We experimented with different number of keywords and keyphrases for the Google query string. The results are presented in Table 2. The best results were obtained with a five keyword, two keyphrase system. We tried increasing the query string length beyond the five keyword, two keyphrase mark, but there was no performance improvement and Google would frequently return a very short list of URLs implying that the query string was too specific.

### 5.3. Effect of rejection model

To study the effectiveness of the rejection model we report results on final system perplexity and vocabulary size with and without the rejection model. As can be seen in Table s4, the rejection model plays an important role in filtering out noisy Webdata thus reducing the system perplexity and keeping out spurious words( web links, Advertisements etc ) from the system vocabulary. The rejection model would typically remove 8% of the downloaded documents and around 6% utterances from the remaining document set in its first iteration. In the subsequent iterations the rejected data size increases and on an average 10% documents and 13% of the utterances are rejected at the end of the download process.

| With Rejection Model | PPL= 92 | Vocab = 45K words |
|---|---|---|
| Without Rejection Model | PPL= 105 | Vocab = 60K words |

**Table 3**. Perplexity(PPL) and vocabulary size of the final system with and without the rejection model.

| Number of weight bins | Perplexity of the final merged LM |
|---|---|
| 10 | 99 |
| 5 | 92 |
| 3 | 103 |
| 0 | 110 |

**Table 4**. Effect of data weighting granularity. The zero bins case is equivalent to having no weighting.

### 5.4. Effect of data weighting

Our final experiments were on the effectiveness of the data weighting strategy. We changed the number of bins in which the downloaded data is pooled and evaluated the effect on system performance. The zero bin case is equivalent to having no data weighting. The optimal performance was seen when the number of bins was kept as five. One possible reason for the performance hit when the number of bins is large is that some of the bins receive little data and the corresponding language model for the bin is poorly estimated.

The gain achieved after merging with Webdata in terms of perplexity is from 117 to 92 or around 20%.

### 5.5. ASR performance

After selecting the best system parameters tuned to minimize perplexity of the heldout text utterances, word error rate experiments were conducted on a test set of 800 utterances(5K words). The baseline interpolated language model consisting of 4000 indomain utterances interpolated with CMU LM. The baseline WER of 28% was reduced to 24% by using the Webdata interpolated LM. This corresponds to a relative gain of 14%.

## 6. CONCLUSION

As our results indicate, acquiring data from the web for building topic specific language models can provide significant improvements in system performance both in terms of perplexity and WER. To maximize the performance improvement that can be achieved from web resources, we need to generate the proper query strings, filter out spurious text and weight the downloaded data properly to reflect its relevance to the topic model we are building.

In this paper, we presented an approach to address these issues by using a competitive set of models. The first model representing generic topic independent data serves as the background model and is used in conjunction with a topic specific model to generate query strings and measure the topic relevance of the downloaded data at both the utterance level and document level. The query generation uses a relative entropy based comparison of the background and topic model to identify word sequences which can serve as keywords and keyphrases. The data is weighted for topic relevance using the likelihood scores computed from the two models.

To identify spurious documents and text in Webdata we built a rejection model from documents and utterances which scored poorly with the background model and topic model. This model was then used to clean the downloaded text in subsequent iterations of web downloads.

We are currently using a simple linear interpolation model for merging the Webdata language model with the existing topic model. Using more complex techniques such as class based model interpolation[7] can help in improving the performance further. Towards that end, we are investigating approaches to generalize our system to gather web data in other languages also.

## 7. ACKNOWLEDGMENTS

This research was funded in parts by the Integrated Media Systems Center, an NSF ERC, under cooperative agreement No. EEC-9529152, by DARPA and by the Department of the Army under contract number DAAD 19-99-D-0046.

## 8. REFERENCES

[1] X. Zhu and R. Rosenfield, “Improving trigram language modeling with the world wide web,” in *Proceedings of ICASSP*, 2001.

[2] Milind Mahajan, Doug Beeferman, and X. D. Huang, “Improved topic-dependent language modeling using information retrieval techniques,” in *Proceedings of ICASSP*, 1999.

[3] A. Berger and R. Miller, “Just-in-time language modeling,” in *Proceedings of ICASSP*, 1998.

[4] Abhinav Sethy, Bhuvana Ramabhadran, and Shrikanth Narayanan, “Measuring convergence in language model estimation using relative entropy,” ICSLP, 2004.

[5] Tim Ng and Mari Ostendorf et al., “Web-data augmented language models for mandarin conversational speech recognition,” in *Proceedings of ICASSP*, 2004.

[6] Ruhi Sarikaya, Agustin Gravano, and Yuqing Gao, “Rapid language model development using external resources for new spoken dialogues domain,” in *Proceedings of ICASSP*, 2004.

[7] Ivan Bulyko, Mari Ostendorf, and Andreas Stolcke, “Getting more mileage from web text sources for conversational speech language modeling using class-dependent mixtures,” in *Proceedings of HLT*, 2003.

[8] N. J. Belkin, D. Kelly, G. Kim, J.-Y. Kim, H.-J. Lee, G. Muresan, M.-C. Tang, X.-J. Yuan, and C. Cool, “Query length in interactive information retrieval,” in *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, 2003.

[9] Andreas Stolcke, “SRILM–an extensible language modeling toolkit,” in *Proceedings of ICSLP*, 2002.

[10] Andrew Kachites McCallum, “Bow: A toolkit for statistical language modeling, text retrieval, classification and clustering,” http://www.cs.cmu.edu/ mccallum/bow, 1996.

[11] S. Narayanan and P. G. Georgiou et al., “Transonics: A speech to speech system for English-Persian interactions,” in *IEEE Automatic Speech Recognition and Understanding Workshop, ASRU*, 2003.

[12] A. Venkataraman and W. Wang, “Techniques for effective vocabulary selection,” in *Proceedings of Eurospeech*, 2003.