Improving Trigram Language Modeling with the World Wide Web

Xiaojin Zhu Ronald Rosenfeld

November 2000
CMU-CS-00-171

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

Abstract

We propose a novel method for using the World Wide Web to acquire trigram estimates for statistical language modeling. We submit an N-gram as a phrase query to web search engines. The search engines return the number of web pages containing the phrase, from which the N-gram count is estimated. The N-gram counts are then used to form web-based trigram probability estimates. We discuss the properties of such estimates, and methods to interpolate them with traditional corpus based trigram estimates. We show that the interpolated models improve speech recognition word error rate significantly over a small test set.

Keywords: Language models, Speech recognition and synthesis, Web-based services

1 Introduction

A language model is a critical component for many applications, including speech recognition. Enormous effort has been spent on building and improving language models. Broadly speaking, this effort develops along two orthogonal directions: The first direction is to apply increasingly sophisticated estimation methods to a fixed training data set (corpus) to achieve better estimation. Examples include various interpolation and backoff schemes for smoothing, variable length N-grams, vocabulary clustering, decision trees, probabilistic context free grammar, maximum entropy models, etc [1]. We can view these methods as trying to "squeeze out" more benefit from a fixed corpus. The second direction is to acquire more training data. However, automatically collecting and incorporating new training data is non-trivial, and there has been relatively little research in this direction. Adaptive models are examples of the second direction. For instance, a cache language model uses recent utterances as additional training data to create better N-gram estimates. The recent rapid development of the World Wide Web (WWW) makes it an extremely large and valuable data source. Just-in-time language modeling [2] submits previous user utterances as queries to WWW search engines, and uses the retrieved web pages as unigram adaptation data. In this paper, we propose a novel method for using the WWW and its search engines to derive additional training data for N-gram language modeling, and show significant improvements in terms of speech recognition word error rate.

The rest of the paper is organized as follows. Section 2 gives the outline of our method, and discusses the relevant properties of the WWW and search engines. Section 3 investigates the problem of combining a traditional corpus with data from the web. Section 4 presents our experimental results. Finally Section 5 discusses both the potential and the limitations of our proposed method, and lists some possible extensions.

2 The WWW as trigram training data

The basic problem in trigram language modeling is to estimate $p(w_3|w_1, w_2)$, i.e. the probability of a word given the two words preceding it. This is typically done by smoothing the maximum likelihood estimate

$$\hat{p}_{ML}(w_3|w_1, w_2) = \frac{c(w_1 w_2 w_3)}{c(w_1 w_2)}$$

with various methods, where $c(w_1 w_2 w_3)$ and $c(w_1 w_2)$ are the counts of "$w_1 w_2 w_3$" and "$w_1 w_2$" in some training data respectively. The main idea behind our method is to obtain the counts of "$w_1 w_2 w_3$" and "$w_1 w_2$" as they appear on the WWW, to estimate

$$\hat{p}_{web}(w_3|w_1, w_2) = \frac{c_{web}(w_1 w_2 w_3)}{c_{web}(w_1 w_2)}$$

and combine $\hat{p}_{web}$ with the estimates from a traditional corpus (here and elsewhere, when $c_{web}(w_1 w_2) = 0$, we regard $\hat{p}_{web}(w_3|w_1, w_2)$ as unavailable). Essentially, we are using the searchable web as additional training data for trigram language modeling.

There are several questions to be addressed. First, how to obtain the counts from the web? What is the quality of these web estimates? How could they be used to improve language modeling? We will examine these questions in the following sections, in the context of N-best list rescoring for speech recognition.

2.1 Obtaining N-gram counts from the WWW

To obtain the count of an N-gram "$w_1 \ldots w_n$" from the web, we use the 'exact phrase search' function of web search engines. That is, we send "$w_1 \ldots w_n$" as a single quoted phrase query to a search engine. Ideally, we would like the search engine to report the phrase count, i.e. the total number of occurrences of the phrase in all its indexed web pages. However in practice, most search engines only report the web page count, i.e. the number of web pages containing the phrase. Since one web page may contain one or more occurrence of the phrase, we need to estimate the phrase count from the web page count.

Many web search engines claim they can perform exact phrase search. However, most of them seem to use an internal stop word list to remove common words from a query phrase. An interesting test phrase is "to be or not to be": Some search engines return totally irrelevant web pages for this query, since most, if not all, words are ignored. In addition, a few search engines perform stemming so the query "she say" will return some web pages only containing "she says" or "she said". Furthermore, some search engines report neither phrase counts nor web page counts. We experimented with a dozen popular search engines, and found three that meet our criteria: AltaVista [3] advanced search mode, Lycos [4], and FAST [5] [1]. They all report web page counts.

One brute force method to get the phrase counts is to actually download all the web pages the search engine finds. However, queries of common words typically result in tens of thousands of web pages, and this method is clearly infeasible. Fortunately at the time of our experiment AltaVista had a simple search mode, which reported both the phrase count and the web page count for a query. Figure 1 shows the phrase count vs. web page count for 1200 queries. Trigram queries (phrases consisting of three consecutive words), bigram queries and unigram queries are plotted separately. There are horizontal branches in the bigram and trigram plots that don't make sense (more web pages than total phrase counts). We regard these as outliers due to idiosyncrasies of the search engine, and exclude them from further consideration. The three plots are largely log-linear. This prompted us to perform the following log-linear regression separately for trigrams, bigrams, and unigrams:

$$c = \alpha_0 * pg^{\alpha_1}$$

where c is the phrase count, and pg the web page count. Table 1 lists the coefficients. The three regression functions are also plotted in Figure 1. We assume these functions apply to other search engines as well. In the rest of the paper, all web N-gram counts are estimated by applying the corresponding regression function to the web page counts reported by search engines.

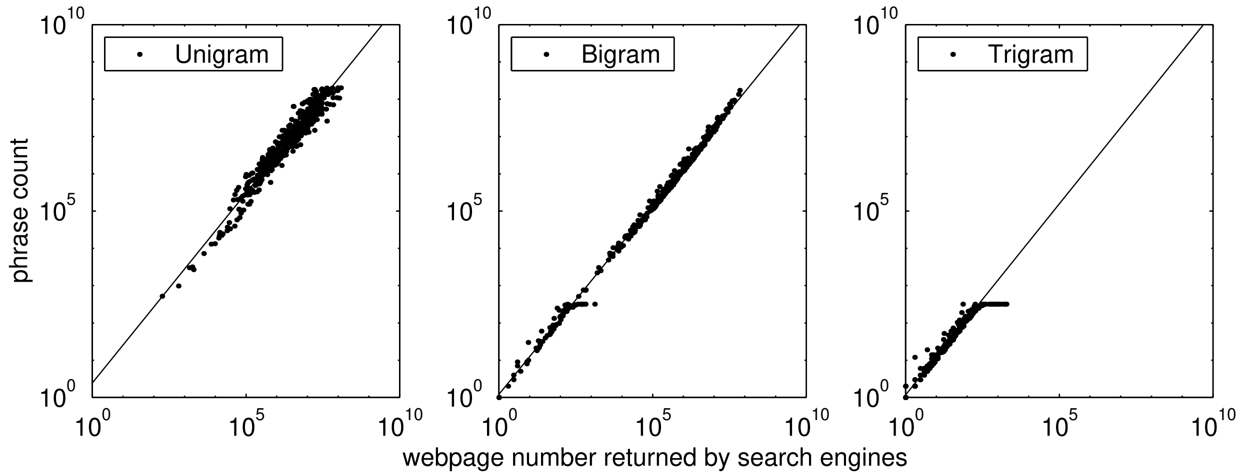

Figure 1: Web phrase count vs. web page count

[1] Our selection is admittedly incomplete. In addition, since search engines develop and change rapidly, all our comments are only valid during the period of this experiment.

	α_0	α_1
Unigram	2.427	1.019
Bigram	1.209	1.014
Trigram	1.174	1.025

Table 1: Coefficients of log-linear regression for estimating Web N-gram counts from Web page counts reported by search engines.

2.2 The quality of web estimates

To investigate the quality of web estimates, we needed a baseline corpus for comparison. The baseline we used is a 103 million word Broadcast News corpus.

2.2.1 Web N-gram coverage

The first experiment we ran was N-gram coverage test on unseen text. That is, we wanted to see how many N-grams in the test text are not on the web, and/or not in the baseline corpus. We were hoping to show that the web covers many more N-grams than the baseline corpus. Note that by 'the web' we mean the searchable portion of the web as indexed by the search engines we chose.

The unseen news test text consisted of 24 randomly chosen sentences from 4 web news sources (CNN, ABC, Fox, BBC) and 6 categories (world, domestic, technology, health, entertainment, politics). All the sentences were selected from the day's news stories, on the day the experiment was carried out. This was to make sure that the search engines hadn't had the time to index the web pages containing these sentences. After the experiment was completed, we checked each sentence, and indeed none of them were found by the search engines yet. Therefore the test text is truly unseen to both the web search engines and the baseline corpus. (The test text is of written news style, which might be slightly different from the broadcast news style in the baseline corpus.)

There are 327 unigram types (i.e. unique words), 462 bigram types and 453 trigram types in the test text. Table 2 lists the number of N-gram types not covered by the different search engines and the baseline corpus, respectively.

	Unique Types	Not Covered By			
		AltaVista	Lycos	FAST	Corpus
Unigram	327	0	0	0	8
Bigram	462	4	5	5	68
Trigram	453	46	46	46	189

Table 2: Novel N-gram types in 24 news sentences

Clearly, the web's coverage, under any of the search engines, is much better than that of the baseline corpus. It is also worth noting that for this test text, any N-gram not covered by the web was also not covered by the baseline corpus.

In the next experiment, we focused on the trigrams in the test text to answer the question "if one randomly picks a trigram from the test text, what's the chance the trigram has appeared c times in the training data?" Figure 2 shows the comparison, with the training data being the baseline corpus and the web through

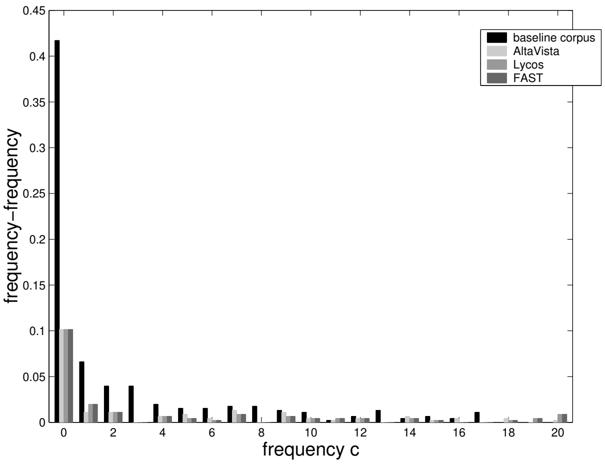

Figure 2: Empirical frequency-frequency plot

the different search engines, respectively. This figure is also known as a "frequency-of-frequency" plot. According to this figure, a trigram from the test text has more than 40% chance of being absent in the baseline corpus, and the chance goes down to about 10% on the web, regardless of the search engine. This is consistent with Table 2. Moreover, the trigram has a much larger chance in having a small count in the baseline corpus than on the web. Since small counts usually mean unreliable estimates, resorting to the web could be beneficial.

2.2.2 The effective size of the web

Recently, Fienberg et al. [6] estimated the size of the indexable web as of 1997 to be close to 1 billion pages. The web grows exponentially, and as of this writing some search engines claim they have indexed more than 1 billion pages. We would like to estimate the effective size of the web as a language model training corpus.

Let's assume that the web and the baseline corpus are homogeneous (which is patently false, since the web has much more than news, but we will ignore this for the time being). Then the probability of a particular N-gram appearing in the baseline corpus is the same as the probability that it appears on the web:

$$p_{corpus}(\text{N-gram}) = p_{web}(\text{N-gram})$$

Since the probabilities can be approximated by their respective frequencies, we have

$$\frac{c_{corpus}(\text{N-gram})}{|\text{corpus}|} \approx \frac{c_{web}(\text{N-gram})}{|\text{web}|}$$

, from which we can estimate $|\text{web}|$, the size of the web in words. Note that it doesn't matter if the N-gram is a unigram, bigram or trigram, though N-grams with small counts are unreliable and should be excluded. In our experiment, we considered all unigrams, bigrams and trigrams in the test text with $c_{corpus} > 10$. Each such N-gram will gave us an estimate, and we took the median of all these estimates for robustness. Table 3 gives our estimates of the size with different search engines.

Some points to notice:

1. The 'effective web size' estimates we obtained are very rough at best. Moreover, they are defined relative to the specific baseline corpus and specific test set we happened to choose. Therefore, Table 3 should not be used to rank the performance of individual search engines.

	Effective size of the web
AltaVista	108 billion words
Lycos	79 billion words
FAST	83 billion words

Table 3: The effective size of the web for language model training

2. This method tends to underestimate the web size. We assumed homogeneity, which in actuality does not hold. The test text comes from a news domain, and so does the baseline corpus. We used N-grams from the test text to estimate the web size, which gives rise to a selectional bias. Intuitively, only "news terms" are in the test text. And since the corpus is in news domain, as a whole we have $p_{corpus}(\text{news terms}) > p_{web}(\text{news terms})$. This is what leads to underestimation.

2.2.3 Normalization of the web counts

An interesting sanity check is to see whether

$$c_{web}(w_1 w_2) = \sum_{w_3 \in \mathrm{V}} c_{web}(w_1 w_2 w_3)$$

holds for any bigram "$w_1 w_2$". If this is true, the relative frequency estimation $\hat{p}_{web}(w_3|w_1, w_2)$ would already be normalized, i.e.

$$\sum_{w_3 \in \mathrm{V}} \hat{p}_{web}(w_3|w_1, w_2) = 1, \forall w_1, w_2$$

Of course there are too many "$w_1 w_2 w_3$" combinations to verify this directly. Instead, we randomly chose six "$w_1 w_2$" pairs from the baseline corpus. For each pair, we chose 2000 w_3's according to the following heuristic: First, we selected words from a list of all w_3's such that the trigram "$w_1 w_2 w_3$" appeared in the baseline corpus, sorted by decreasing frequency; If fewer than 2000 words were chosen that way, we added words from a list of all w_3's such that the bigram "$w_2 w_3$" appeared in the baseline corpus, in decreasing frequency order; If this was still not enough, we added w_3's according to their unigram frequencies. We expected this heuristic to give us a list of w_3's that covers the majority of the conditional probability mass given history "$w_1 w_2$".

Table 4 shows web bigram count estimates obtained with FAST search, together with their respective cumulative web trigram count estimates as described above. Ideally, the ratio should be close to, but less than, 100%. It is evident from the table that the web counts are not perfectly normalized. The reasons are not entirely clear to us, but the fact that the N-gram counts are estimated from page counts is an obvious candidate. The web N-gram count estimates should therefore be used with caution.

2.2.4 The variance and bias of web trigram estimates

As stated earlier, we are interested in estimating conditional trigram probabilities based on their relative frequency on the web:

$$\hat{p}_{web}(w_3|w_1, w_2) = \frac{c_{web}(w_1 w_2 w_3)}{c_{web}(w_1 w_2)}$$

It would be informative to compare $\hat{p}_{web}(w_3|w_1, w_2)$ to a traditional (corpus derived) trigram probability estimate.

"$w_1 w_2$"	$c_{web}(w_1 w_2)$	$\sum_{2000\ w_3\text{'s}} c_{web}(w_1 w_2 w_3)$	ratio
about seventy	16498.3	14807.7	90%
and there's	662697.0	724870.0	109%
group being	20248.4	16246.5	80%
lewinsky after	1431.9	1631.7	114%
two hundred	389949.0	457656.0	117%
willy b.	1334.6	607.2	45%

Table 4: Sanity check: are web counts normalized?

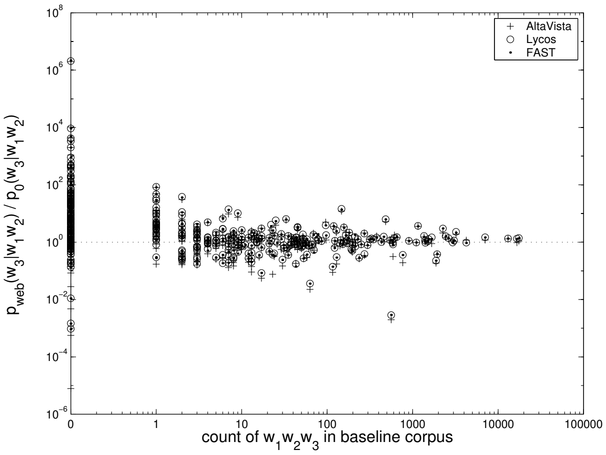

Figure 3: Ratio of web trigram estimates to corpus trigram estimates

To this end, we created a baseline trigram language model LM_0 from the 103 million word baseline corpus. We used modified Kneser-Ney smoothing [7] [8] which, according to [8], is one of the best smoothing methods available. In building LM_0, we discarded all singleton trigrams in the baseline corpus, a common practice to reduce language model size. We denote LM_0's probability estimates by p_0.

With LM_0, we were able to compare $\hat{p}_{web}(w_3|w_1, w_2)$ with $p_0(w_3|w_1, w_2)$. We computed the ratio r:

$$r(w_1, w_2, w_3) = \frac{\hat{p}_{web}(w_3|w_1, w_2)}{p_0(w_3|w_1, w_2)}$$

between these two estimates. We expected r to be more spread out (having larger variance) when $c_{corpus}(w_1 w_2 w_3)$ is small, since in this case $p_0(w_3|w_1, w_2)$ tends to be unreliable.

We computed $r(w_1, w_2, w_3)$ for every trigram in the test text, excluding those with $c_{web}(w_1 w_2) = 0$. We plot $r(w_1, w_2, w_3)$ vs. $c_{corpus}(w_1 w_2 w_3)$ in Figure 3. We found that:

1. For trigrams with large $c_{corpus}(w_1 w_2 w_3)$, r averages to about 1. Thus the web estimates are consistent with LM_0 in this case.

2. As we expected, the variance of r is largest when $c_{corpus}(w_1 w_2 w_3)$=0, and decreases when it gets large. Hence the 'funnel' shape.

3. When $c_{corpus}(w_1 w_2 w_3)$ is small, especially 0 and 1, r is biased upward. This is of course good news, as it suggests that this is where the web estimates tend to improve on the corpus estimates.

4. All the search engines give similar results.

3 Combining web estimates with existing language model

In the previous section, we saw the potential of the web: it is huge, it has better trigram coverage, and its trigram estimates are largely consistent with the corpus-based estimates. Nevertheless, to query each and every N-gram on the web is infeasible. This prevents us from building a full fledged language model from the web via search engines. More over, Table 4 indicates that web estimates are not well normalized. In addition, the content of the web is heterogeneous and usually doesn't coincide with our domain of interest. Based on these considerations, we decided not to try to build an entire language model from the web. Rather, we will start from a traditional language model LM_0, and interpolate its least reliable trigram estimates with the appropriate estimates from the web.

Unreliable trigram estimates, especially those involving backing off to lower order N-grams, have been shown to be correlated with increased speech recognition errors [9] [10]. By going to the much larger web for reliable estimates, Our hope was to alleviate this problem. We used the trigram counts in the baseline corpus as a heuristic to decide the reliability of trigram estimates in LM_0. A trigram estimate $p_0(w_3|w_1, w_2)$ is deemed unreliable, if

$$c_{corpus}(w_1 w_2 w_3) \leq \tau$$

where τ, the 'reliability threshold', is a predetermined small positive integer, e.g. 1. Admittedly this definition of unreliable estimates is biased.

Even with this definition, there are still too many unreliable trigram estimates to query the web for. Since we were interested in N-best list rescoring, we further restricted the queries to those unreliable trigrams that appeared in the particular N-best list being processed. This greatly reduces the number of web queries at the price of some further bias. Let $U_{w_1 w_2}$ be the set of words that have unreliable trigram estimates with history "$w_1 w_2$" in the current N-best list, i.e.

$$\begin{aligned} &\forall \text{"} w_1 w_2 \text{"} \in \text{N-best} \wedge c_{web}(w_1 w_2) > 0, \\ U_{w_1 w_2} &= \{w_3 | \text{"} w_1 w_2 w_3 \text{"} \in \text{N-best} \wedge c_{corpus}(w_1 w_2 w_3) \leq \tau\} \end{aligned} \tag{1}$$

We obtain $c_{web}(w_1, w_2, u), u \in U_{w_1 w_2}$ and $c_{web}(w_1 w_2)$ via search engines, and compute $\hat{p}_{web}(u|w_1, w_2)$, the web relative frequency estimates, from these web counts.

Let $p^*(u|w_1, w_2)$ denote the final interpolated estimates, which combine $p_0(u|w_1, w_2)$ and $\hat{p}_{web}(u|w_1, w_2)$. We would like to have a tunable parameter so that on one extreme $p^*(u|w_1, w_2) \to p_0(u|w_1, w_2)$, while on the other extreme $p^*(u|w_1, w_2) \to \hat{p}_{web}(u|w_1, w_2)$. We now present three different methods for doing this.

3.1 Exponential Models with Gaussian Priors

We define a set of binary functions, or 'features', as follows:

$$f_{w_1, w_2, u}(w_3) = \begin{cases} 1 & \text{if } u = w_3 \\ 0 & \text{otherwise} \end{cases}$$

for all $w_1, w_2, u \in U_{w_1 w_2}$ in the N-best list. Next, for any given w_1, w_2, we define a conditional exponential model p^*_E with these features:

$$\begin{aligned} &p^*_E(w_3|w_1, w_2) = \\ &\tfrac{1}{Z_{w_1 w_2}} p_0(w_3|w_1, w_2) \exp\left(\textstyle\sum_{u \in U_{w_1 w_2}} \lambda_u f_{w_1, w_2, u}(w_3)\right) \end{aligned} \tag{2}$$

where p_0 is the estimate provided by LM_0, λ's are parameters to be optimized, and $Z_{w_1 w_2}$ is a normalization factor. This model has exactly the same form as a conventional Maximum Entropy / Minimum Discriminative Information (ME/MDI) model [11] [12]. Let Λ denote the set of parameters. If we maximize the likelihood of the web counts:

$$L(\Lambda) = \prod_{w_1, w_2, w_3} p_E^*(w_3|w_1, w_2)^{c_{web}(w_1 w_2 w_3)}$$

with the standard Generalized Iterative Scaling algorithm (GIS) [13], we get the ME/MDI solution that satisfies the following constraints:

$$\begin{aligned} p_E^*(u|w_1, w_2) &= \hat{p}_{web}(u|w_1, w_2) \\ &= \frac{c_{web}(w_1, w_2, u)}{c_{web}(w_1, w_2)}, \forall w1, w2, u \in U_{w_1 w_2} \end{aligned} \tag{3}$$

This corresponds to one extreme of the interpolation. But since we want to control the degree of interpolation, we introduce a Gaussian prior with mean 0 and variance σ^2 over Λ:

$$p(\Lambda) = \prod_i \frac{1}{\sqrt{2\pi\sigma^2}} \exp(\frac{-\lambda_i^2}{2\sigma^2})$$

And instead of seeking the maximum likelihood solution, we seek the maximum a posteriori (MAP) solution that maximizes

$$L(\Lambda) * p(\Lambda)$$

This can be done by slightly modifying the GIS algorithm, as described in [14].

With this Gaussian prior, we can control the degree of interpolation by choosing the value of $\sigma^2 \in (0, +\infty)$. σ^2 acts as a tuning parameter: If $\sigma^2 \to +\infty$, the Gaussian prior is flat and has virtually no restriction on the values of the λ's. Thus the λ's can reach their ME/MDI solutions, and hence p_E^* reaches one extreme as in (3). On the other hand if $\sigma^2 \to 0$, the Gaussian prior forces λ's to be close to the mean, which is 0. From (2) we know in this case $p_E^* \to p_0$. This corresponds to the other extreme of the interpolation. A σ^2 between 0 and $+\infty$ results in an intermediate p_E^* distribution.

For the purpose of comparison, we experimented with two other interpolation methods, which are easy to implement but may be theoretically less well motivated: linear interpolation and geometric interpolation.

3.2 Linear Interpolation

In linear interpolation, we have

$$p_L^*(w_3|w_1, w_2) = \begin{cases} (1-\alpha)p_0(w_3|w_1, w_2) + \alpha\hat{p}_{web}(w_3|w_1, w_2) \\ \qquad\qquad , \text{if } w_3 \in U_{w_1 w_2} \\ \frac{1-\sum_{u \in U_{w_1 w_2}} p_L^*(u|w_1, w_2)}{1-\sum_{u \in U_{w_1 w_2}} p_0(u|w_1, w_2)} p_0(w_3|w_1, w_2) \\ \qquad\qquad , \text{otherwise} \end{cases} \tag{4}$$

In this case, $\alpha \in [0, 1]$ is the tuning parameter. If $\alpha = 0$, $p_L^* = p_0$. If $\alpha = 1$, p_L^* satisfies (3). An α in between results in an intermediate p_L^*.

3.3 Geometric Interpolation

In geometric interpolation, we have

$$p_G^*(w_3|w_1, w_2) = \tag{5}$$
$$\begin{cases} p_0(w_3|w_1, w_2)^{(1-\beta)} \left[\frac{c_{web}(w_1 w_2 w_3)+\epsilon}{c_{web}(w_1 w_2)+|V|\epsilon}\right]^{\beta} & \text{, if } w_3 \in U_{w_1 w_2} \\ \frac{1-\sum_{u \in U_{w_1 w_2}} p_G^*(u|w_1,w_2)}{1-\sum_{u \in U_{w_1 w_2}} p_0(u|w_1,w_2)} p_0(w_3|w_1, w_2) & \text{, otherwise} \end{cases}$$

Note that here we have to smooth the web estimates to avoid zeros (which is not a problem in the previous two methods). To do this, we simply add a small positive value ϵ to the web counts. This is known as additive smoothing [8]. The value of ϵ is determined to minimize the perplexity with $\beta = 1$. Once ϵ is chosen it is fixed, and we tune β. $\beta \in [0, 1]$ is the interpolation parameter. If $\beta = 0$, $p_G^* = p_0$. If $\beta = 1$, p_G^* satisfies the smoothed web estimates. A β in between results in an intermediate p_G^*.

4 Experimental Result

We randomly selected 200 utterance segments from the TREC-7 Spoken Document Retrieval track data [15] as our test set for this experiment. For each utterance we have its correct transcript and an N-best list with $N = 1000$, i.e. 1000 decoding hypotheses. We performed N-best list rescoring to measure the word error rate (WER) improvement, and computed the perplexity of the transcript. Note that the test set is relatively small and $N = 1000$ is not very deep, since we wanted to limit the number of web queries to within a practical range.

4.1 Word Error Rate

If we rescore the N-best lists with LM_0 and pick the top hypotheses, the WER is 33.45%. This is our baseline WER. The oracle WER, i.e. if we were able to pick the least errorful hypothesis among the 1000 for each N-best list, is 25.26%. Of course we cannot achieve the oracle WER, but it indicates there is room for improvement over LM_0.

Since each utterance has 1000 hypotheses in the N-best list, the total number of trigrams is very large. Table 5 lists the number of trigram tokens (occurrences) and types (unique ones) in all the N-best lists combined, together with the percentage of unreliable trigram types and tokens as determined by the reliability threshold τ. Note that trigrams containing start-of-sentence or end-of-sentence (commonly designated by $< s >$ and $< /s >$) are excluded from the table, since they can't be queried from the web. For each N-best list, we queried the unreliable trigrams (and associated bigrams) in the list, from which we computed p^* with the three different interpolation methods. We then used p^* to rescore the N-best list, and calculated the WER of the top hypothesis after rescoring.

First, we set the reliability threshold $\tau = 0$, i.e. we regard only those trigrams that never occur in the baseline corpus as unreliable. Figure 4(a) shows the WER with exponential models and Gaussian priors. The three curves stand for different search engines, which turn out to be very similar. The horizontal dashed line is the baseline WER. As predicted, when the variance of the Gaussian prior $\sigma^2 \to 0$ (the left side of the figure), p_E^* converges to p_0 and the WER converges to the baseline WER. On the other hand when $\sigma^2 \to +\infty$, the estimates of the unreliable trigrams come solely from the web. Such estimates seem inferior

trigram	total	reliability threshold τ					
		0	1	2	3	4	5
tokens	5,311,303	2,002,530 37.7%	2,310,416 43.5%	2,496,312 47.0%	2,650,340 49.9%	2,772,500 52.2%	2,889,348 54.4%
types	57,107	36,190 63.4%	39,059 68.4%	40,893 71.6%	42,158 73.8%	43,110 75.5%	43,863 76.8%

Table 5: Number of unreliable trigrams in the N-best lists

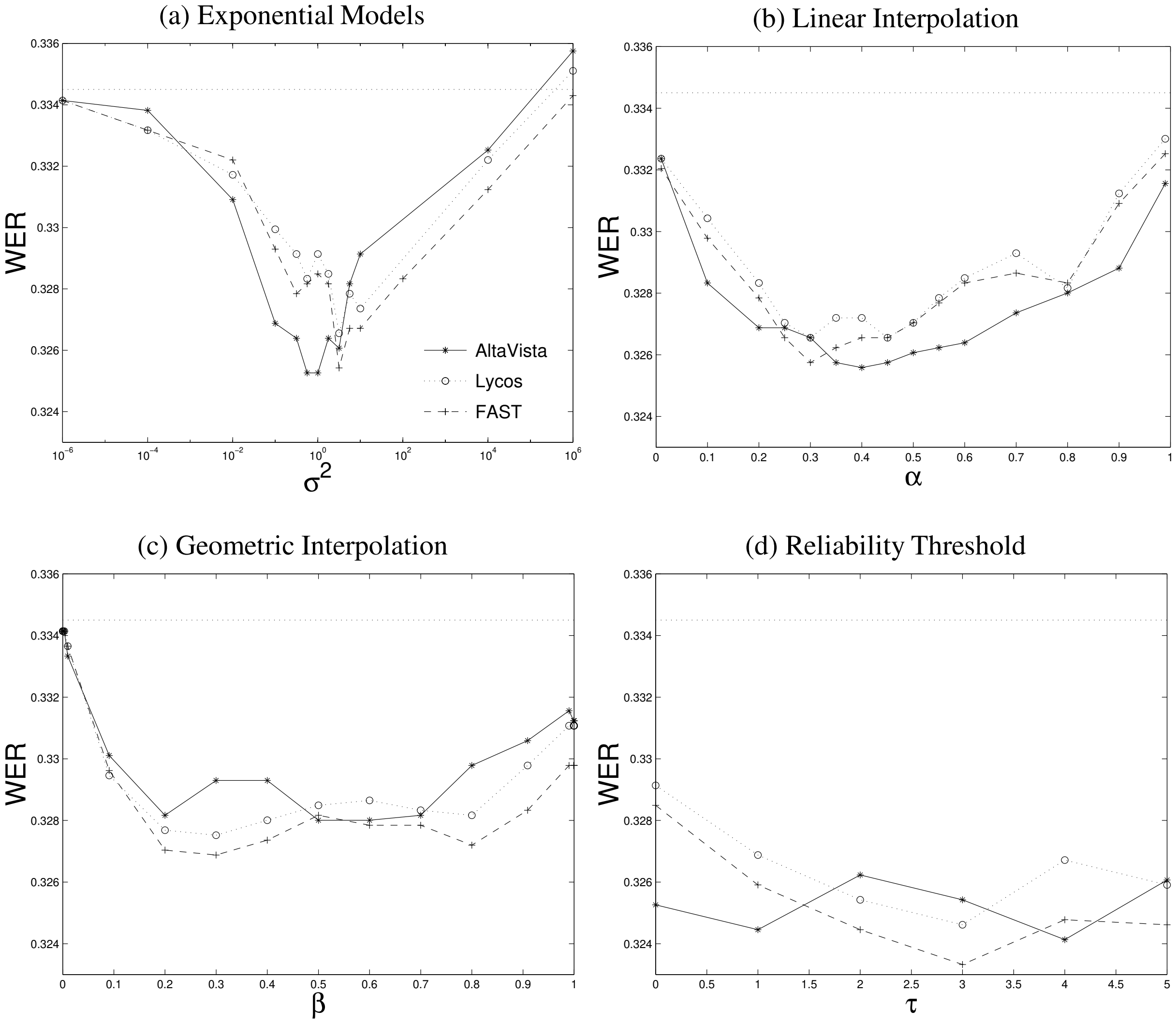

Figure 4: Word Error Rates of web-improved language models as function of the smoothing parameter for several different interpolation schemes, based on N-best rescoring

and the model has higher WER than the baseline. Between these two extremes, WER reaches minimum (32.53% with AltaVista) around $\sigma^2 = 1$.

Figure 4(b) is the WER with linear interpolation. Again, the minimum WER 32.56% is reached between the two extremes at $\alpha = 0.4$ by AltaVista.

To use geometric interpolation, we needed to choose a value for ϵ first. We chose $\epsilon = 0.01$ because this minimized the perplexity when $\beta = 1$. Next we vary β while keeping ϵ fixed, and plotted the WER of the interpolated model in Figure 4(c). As with the previous interpolation methods, the WER reaches minimum when the interpolation factor is near the middle. The minimum is 32.69% when $\beta = 0.3$ with FAST.

Next, we adjusted the reliability threshold τ and observe its effect on WER. The interpolation method used here is the exponential model with Gaussian prior and $\sigma^2 = 1$. We varied τ from 0 to 5. With larger threshold, more trigrams are regarded as unreliable, and hence more web queries had to be issued. As shown in Figure 4(d), there is a slight but definite improvement in WER when we increase τ from 0 to 1. For example, The WER with $\tau = 1$ and AltaVista is 32.45%. Further increment results in about the same WER, averaged over search engines. Note that LM_0, the language model we are incorporating web estimate into, was built after excluding all singleton trigrams in the corpus. This may explain why $\tau = 1$ is better since trigrams with counts 0 or 1 in the corpus are indeed unreliable: in LM_0 they must backoff to bigram or unigram.

To analyze the source of improvement, we broke down the WER according to the trigram backoff modes in LM_0. First, we marked each word w_i in the transcript with one of several labels, using the following rules: Let w_{i-2} and w_{i-1} be the two words preceding w_i. If the trigram "$w_{i-2}w_{i-1}w_i$" exists in LM_0, label w_i as '3'. Otherwise if the trigram doesn't exist in LM_0, but the bigram "$w_{i-1}w_i$" does, label w_i as '3-2', meaning LM_0 has to backoff to the bigram for w_i. If the bigram doesn't exist in LM_0 either, label w_i as '3-2-1' since LM_0 has to backoff to the unigram. In the second step, we compared the transcript with the top hypotheses after rescoring the N-best lists with p_0. Each word in the transcript obtains a second label of either "correct" or "wrong" depending on whether the word is correct in the corresponding top hypothesis. We then collect the percentage of correct words within categories '3', '3-2' and '3-2-1' respectively. In the third step we repeated the second step, except that the top hypotheses are now obtained by rescoring the N-best lists with p_E^*, where $\sigma^2 = 1$, $\tau = 1$, and the search engine is AltaVista. We compare the percentage of errors in step 2 and step 3 in Table 6. Note that insertion errors are not counted in our error break down. Not surprisingly, the '3-2-1' category has the highest error rate for both p_0 and p_E^*, since the words in this category are the hardest from the language model's point of view. The '3-2' category has lower error rate, and '3' has the lowest. The interpolated language model p_E^* improves error rate for all three categories, compared to p_0. The largest improvement is in the '3-2-1' category, which suggests the web helps LM_0 most with the hardest cases. It is not clear though why the '3-2' category is not improved as much.

category	words	error rate p_0	error rate p_E^*
3	3480	23.3%	22.8%
3-2	2236	30.7%	30.1%
3-2-1	479	50.1%	46.1%

Table 6: Error break down by LM_0 backoff mode

4.2 Approximate Perplexity

There are 6195 words in the transcript. The baseline perplexity of the transcript with LM_0 is 196.7. We wanted to compute the perplexity of the transcript with different interpolated language models. We define $U_{w_1 w_2}$ in (1) based on the transcript. However this introduces a subtle bias: the interpolated models now depend on the transcript. In other words, we are dynamically choosing models according to the words we will be predicting. The resulting scores are therefore not strictly interpretable as probabilities. For this reason we consider the perplexities we get on the transcript to be approximate only. We still report these values in this section because we believe that the distortion is not too severe, and the approximation still provides useful insight into the true perplexity of web-improved language models. Note that, although the same kind of bias exists in WER computation, it doesn't diminish the validity of the WER improvement we get there, since in classification it is not the particular probability value but the ranking that matters.

Figure 5(a–c) compares different interpolation methods when the reliability threshold $\tau = 0$. There are 2274 unique unreliable trigrams in the transcript. We submitted them (and the corresponding bigrams) as queries to the search engines, and computed p^* with the three different interpolation methods described in the last sections respectively. From p^* we computed the approximate perplexities.

Figure 5(a) shows the approximate perplexity with the exponential model and a Gaussian prior. Like the WER in Figure 4(a), the approximate perplexity converges to the baseline when the Gaussian prior $\sigma^2 \to 0$. The approximate perplexity worsens when $\sigma^2 \to +\infty$. The best value 156.9 is achieved by FAST also between these two extremes at $\sigma^2 = 1$. Again, different search engines are similar.

Figure 5(b) is the approximate perplexity with linear interpolation. It is also similar to the WER in Figure 4(b). The minimum 156.2 is reached by FAST at $\alpha = 0.45$.

Figure 5(c) shows the approximate perplexity with geometric interpolation and $\epsilon = 0.01$. As with the previous interpolation methods, the approximate perplexity converges to the baseline when $\beta \to 0$ and is worse when $\beta \to 1$. But unlike the other methods, approximate perplexity seems to be always worse than the baseline, and increases monotonically with β.

Figure 5(d) compares the effect of the reliability threshold τ on the approximate perplexity. As in Figure 4(d), the interpolation method used is exponential model with Gaussian prior and $\sigma^2 = 1$. Again we see improvement when we increase τ from 0 to 1. For example, FAST's approximate perplexity goes down to 147.5. We believed this can be explained similarly to Figure 4(d).

5 Discussions

In this paper, we demonstrated that trigram estimates obtained from the web can significantly improve WER relative to pure corpus-based estimates, even though the web estimates are noisy, and the web and the test set are not in the same domain. We believe the improvement largely comes from better trigram coverage due to the sheer size of the web, which acts as a 'general English' knowledge source. Interestingly, which search engine is used doesn't make much difference. Furthermore, which interpolation method is used doesn't make much difference either (at least for WER), as long as an appropriate interpolation parameter is chosen.

Our method has certain advantages. Besides having better N-gram coverage, the content of the web is constantly changing. Our method would enable automatic up-to-date language modeling. However, there are also several disadvantages. The most severe one is the large number of web queries. In our experiment, we needed to submit an average of 340 queries to the web for each utterance. This results in heavy web traffic and workload on the search engines, and very slow rescoring process. Another concern is privacy: one may be sending fragments of potentially sensitive utterances to the web. Both problems, however, can be partly solved by using a web-in-a-box setting, i.e. if we have a snapshot of the text content of the whole WWW on local storage. Yet another problem is the lack of focus on domain specific language. This might

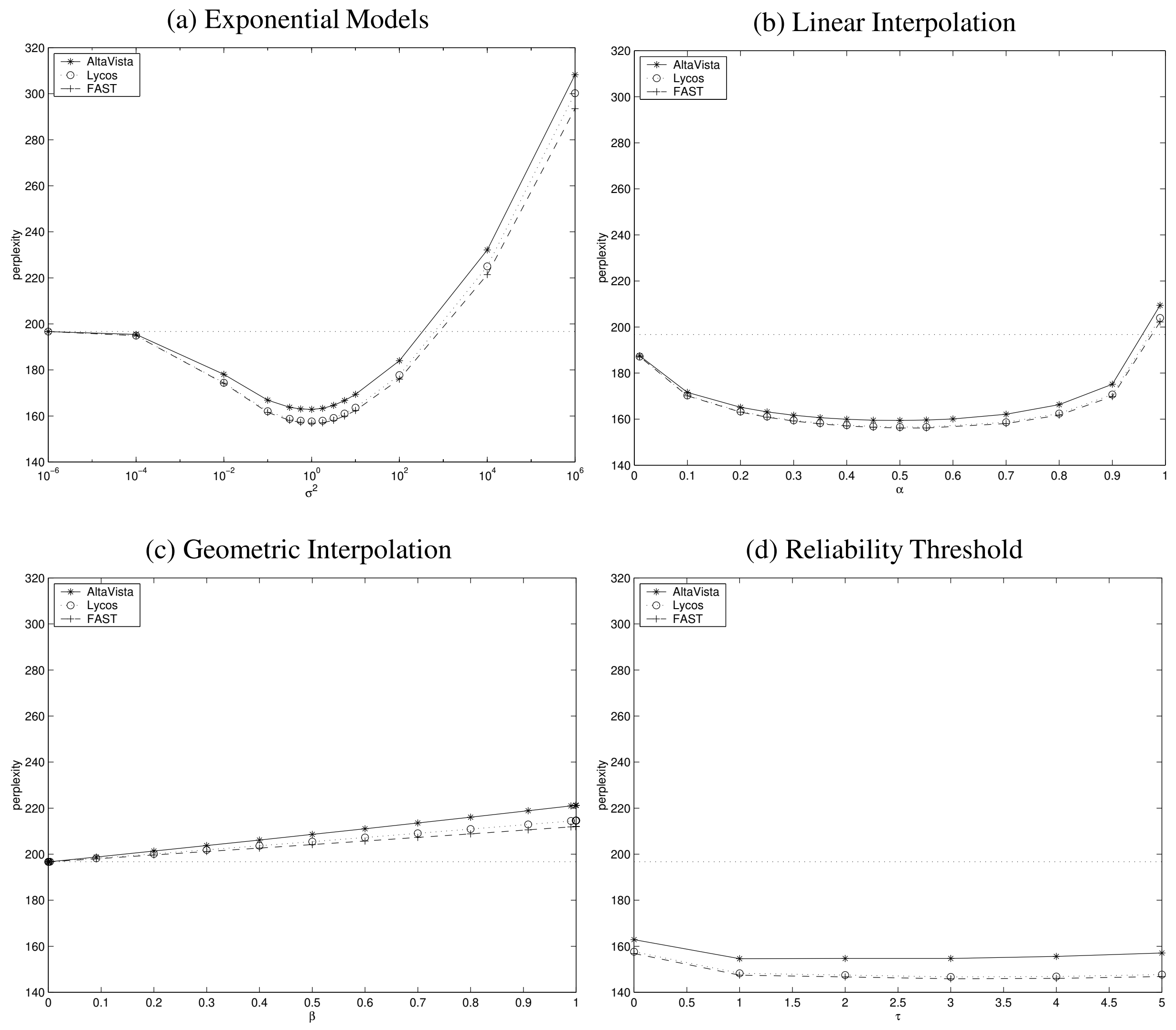

Figure 5: Approximate perplexity of web-improved language models as function of the smoothing parameter for several different interpolation schemes.

be solved by querying specific domain hosts instead of the whole web, although by doing so the N-gram coverage may deteriorate.

The method proposed in this paper is only one crude way of exploiting the web as a knowledge source for language modeling. Instead of focusing on trigrams, one could look for more complex phenomena, e.g. semantic coherence [16] among the content words in a hypothesis. Intuitively, if a hypothesis has content words that 'go with each other', it is more likely than one whose content words seldom appear together in a large training text set. The web + search engine approach seems well suited for this purpose. We are currently pursing this direction.

Acknowledgement

The authors are grateful to Stanley Chen, Matthew Siegler, Chris Paciorek and Kevin Lenzo for their help. The first author has been supported in part by NSF LIS under grant REC-9720374.

References

[1] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8), 2000.

[2] Adam Berger and Robert Miller. Just-in-time language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume II, pages 705–708, Seattle, Washington, 1998.

[3] AltaVista. http://www.altavista.com/.

[4] Lycos. http://www.lycos.com/.

[5] FAST Search. http://www.alltheweb.com/.

[6] Stephen E. Fienberg and Adrian Dobra. How big is the world wide web? Technical report, Department of Statistics, Carnegie Mellon University, 2000. Submitted for publication.

[7] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 181–184, Detroit, Michigan, May 1995.

[8] Stanley F. Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998. Available from `ftp://ftp.das.harvard.edu/techreports/tr-10-98.ps.gz`.

[9] Lin Chase, Ronald Rosenfeld, and Wayne Ward. Error-responsive modifications to speech recognizers: Negative N-grams. In *Proceedings of the ICSLP*, 1994.

[10] Stanley F. Chen, Douglas Beeferman, and Ronald Rosenfeld. Evaluation metrics for language models. In *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, pages 275–280, 1998.

[11] S. Della Pietra, V. Della Pietra, R.L. Mercer, and S. Roukos. Adaptive language modeling using minimum discriminant estimation. In *Proceedings of the Speech and Natural Language DARPA Workshop*, February 1992.

[12] Adam Berger, Stephen Della Pietra, and Vincent Della Pietra. A maximum entropy approach to natural language processing. *Computational Linguistics*, 22(1):39–71, 1996.

[13] J.N. Darroch and D. Ratcliff. Generalized iterative scaling for log-linear models. *The Annals of Mathematical Statistics*, 43:1470–1480, 1972.

[14] Stanley F. Chen and Ronald Rosenfeld. A Gaussian prior for smoothing maximum entropy models. Technical Report CMU-CS-99-108, Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, 1999.

[15] John S. Garofolo, Ellen M. Voorhees, Cedric G. P. Auzanne, Vincent M. Stanford, and Bruce A. Lund. 1998 TREC-7 spoken document retrieval track overview and results. In *Proceedings of TREC-7: The Seventh Text Retrieval Conference*, 1998.

[16] Can Cai, Larry Wasserman, and Roni Rosenfeld. Exponential language models, logistic regression, and semantic coherence. In *Proceedings of the NIST/DARPA Speech Transcription Workshop*, May 2000.