Learning N-gram Language Models from Uncertain Data

Vitaly Kuznetsov[1,2]*, Hank Liao*[2]*, Mehryar Mohri*[1,2]*, Michael Riley*[2]*, Brian Roark*[2]
[1]Courant Institute, New York University
[2]Google, Inc.
{vitalyk,hankliao,mohri,riley,roark}@google.com

Abstract

We present a new algorithm for efficiently training n-gram language models on uncertain data, and illustrate its use for semi-supervised language model adaptation. We compute the probability that an n-gram occurs k times in the sample of uncertain data, and use the resulting histograms to derive a generalized Katz back-off model. We compare three approaches to semi-supervised adaptation of language models for speech recognition of selected YouTube video categories: (1) using just the one-best output from the baseline speech recognizer or (2) using samples from lattices with standard algorithms versus (3) using full lattices with our new algorithm. Unlike the other methods, our new algorithm provides models that yield solid improvements over the baseline on the full test set, and, further, achieves these gains without hurting performance on any of the set of video categories. We show that categories with the most data yielded the largest gains. The algorithm has been released as part of the OpenGrm n-gram library [1].

1. Introduction

Semi-supervised language model adaptation is a common approach adopted in automatic speech recognition (ASR) [2, 3, 4, 5, 6]. It consists of leveraging the output of a speech recognition system to adapt an existing language model trained on a source domain to a target domain for which no human transcription is available. For example, initial language models for voice search applications can be trained largely on typed queries, then later adapted to better fit the type of queries submitted by voice using the ASR output for large quantities of spoken queries [7]. Another related scenario is that of off-line recognition of large collections of audio or video, such as lectures [8, 5] or general video collections such as those on YouTube [9]. In these cases, some degree of self-adaptation or transductive learning can be carried out on the recordings by folding the output of ASR back into language model training and re-recognizing. Most often, one-best transcripts – possibly with some confidence thresholding – are folded back in for adaptation [6, 5, 10]. In this paper, we investigate methods for adaptation of language models using uncertain data, in particular the full lattice output of an ASR system.[1] This is a special instance of the general problem of learning from uncertain data [11].

Adaptation of language models using lattice output was explored in [6], where consistent word-error rate (WER) reductions versus just adapting on one-best transcripts were demonstrated on a voicemail transcription task. Expected frequencies can be efficiently computed from conditionally normalized word lattices, using the algorithm presented in [12], and these can serve as generalized counts for the purpose of estimating maximum likelihood n-gram language models. However, rather than use expected n-gram frequencies, [6] instead employed a brute-force sampling approach, in order to avoid tricky issues in model smoothing with fractional counts. In this paper, we also demonstrate consistent improvements from lattices over using just one-best transcripts to adapt, but, here, we present new algorithms for estimating the models directly from fractional counts derived from expected frequencies, thereby avoiding costly sampling. The algorithms has been released as part of the OpenGrm NGram Library[2] [1].

In what follows, we first review Katz back-off language modeling [13], which is the language modeling choice for this application, due to its good performance both with very large vocabularies and (in contrast to Kneser-Ney smoothing [14], which is otherwise very popular) in scenarios with extensive model pruning [15, 16]. We next present our new algorithm for estimating a generalized Katz back-off language model directly from fractional counts. Finally, we evaluate our methods on recognition of a selection of channel lineups in YouTube. We find that our lattice-based methods provide solid gains over the baseline model, without hurting performance in any of the lineups. In contrast, one-best adaptation yielded no improvements overall, since it hurt performance on some of the lineups.

2. Katz Back-off Models

In this section, we review Katz back-off language models [13]. Let V be a finite set of words, that is the *vocabulary*. We will denote by $w \in V$ an arbitrary word from this vocabulary. We assume that we are provided with a sample S of m sentences drawn i.i.d. according to some unknown distribution, where each sentence is simply a sequence of words from V. The goal is to use this sample S to estimate the conditional probability $\Pr(w|\mathbf{h})$, where $\mathbf{h}w$, an n-gram sequence, is a concatenation of an arbitrary sequence of $n-1$ words $\mathbf{h}$ (the *history*) and a single word w. Katz [13] proposed the following model as an estimator for $\Pr(w|\mathbf{h})$:

$$\widehat{\Pr}(w|\mathbf{h}) = \begin{cases} d_{c_S(\mathbf{h}w)} \frac{c_S(\mathbf{h}w)}{c_S(\mathbf{h})} & \text{if } c_S(\mathbf{h}w) > 0, \\ \beta_{\mathbf{h}}(S)\widehat{\Pr}(w|\mathbf{h}') & \text{otherwise,} \end{cases} \quad (1)$$

where $c_S(\mathbf{h}w)$ denotes the number of occurrences of the sequence $\mathbf{h}w$ in the training corpus S and where $\mathbf{h}'$ is the longest proper suffix of $\mathbf{h}$, which we will denote by $\mathbf{h}' <_{\text{suf}} \mathbf{h}$, thus $\mathbf{h}' = (w_2, \ldots, w_n)$ if $\mathbf{h} = (w_1, \ldots, w_n)$. $\beta_{\mathbf{h}}(S)$ is a parameter obtained by enforcing normalization:

$$\sum_{w \in V} \widehat{\Pr}(w|\mathbf{h}) = 1.$$

To complete the definiton of the model, it remains to specify the *discount factors* d_k for each order $k \in \mathbb{N}$. Let $S_k = \{\mathbf{w} \in$

[1]Note that, for this paper, we are exclusively using speech data for model adaptation and no side information or meta-data.

[2]http://ngram.opengrm.org/

$V^m\colon c_S(\mathbf{w}) = k\}$ and $n_k = |S_k|$. When $k \leq K$ and $m > 1$, d_k is defined as follows:

$$d_k = \frac{(k+1)n_{k+1}}{kn_k}. \tag{2}$$

When $k > K$ (typically $K = 5$) or $m = 1$ (unigrams), there is no discounting: $d_k = 1$; i.e. the maximum likelihood estimate is used. The derivation of the discount factors d_k makes use of the *Good-Turing estimate* as described in Section 3.5.

3. Fractional Counts and Generalized Katz Back-off Models

In this section, we present fractional count language models for the scenario of learning with uncertain data described in Section 1. We assume that instead of a sample S that is just a collection of sentences, we are given a sequence of *lattices* $\mathcal{L}_1, \ldots, \mathcal{L}_m$ drawn from some unknown distribution $\mathcal{D}$. Each lattice $\mathcal{L}_i$, $i = 1, \ldots, m$, is a probability distribution over a finite set of sentences, which may be the set of hypotheses and associated posterior probabilities output by a speech recognizer for a given spoken utterance. Each lattice can be compactly represented as an acyclic weighted finite automaton.

As before, the goal is to use this sample to build a language model. An important requirement is that we should be able to compute the desired solution in an efficient manner. The solution that we propose is based on simple histogram statistics that can be computed efficiently from each lattice $\mathcal{L}_i$.

We first briefly review some alternative approaches to language modeling using uncertain data and then present our fractional count language models.

3.1. One-Best Language Models

A simple approach to dealing with uncertain data consists of extracting the most likely path from each of the lattices $\mathcal{L}_1, \ldots, \mathcal{L}_m$ to obtain a sample $x_1, \ldots, x_m$. However, this approach may ignore other relevant information contained in the full lattices $\mathcal{L}_1, \ldots, \mathcal{L}_m$.

3.2. Monte Carlo Language Models

[6] showed that using information beyond the one-best path can help improve performance. In particular, [6] used Monte Carlo sampling to accomplish this: samples $S_1, \ldots, S_M$ are drawn from the lattice distribution $\mathcal{L}$ which is a concatenation of lattices $\mathcal{L}_1, \ldots, \mathcal{L}_m$ and for each sample a new model $\widehat{p}_{S_i}$ is constructed. The final model is an interpolation of these models.

3.3. Fractional Count Language Models

Fractional count LMs can be viewed as an ensemble of LMs that estimates the asymptotic limit of the sampling procedure described in Section 3.2 directly, without sampling. More precisely, we define the estimate of the probability of a sequence of words $\mathbf{w}$ to be

$$\widehat{\Pr}[\mathbf{w}] = \mathop{\mathbb{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{w})] = \sum_S \Pr_{S\sim\mathcal{L}}[S]\widehat{p}_S(\mathbf{w}), \tag{3}$$

where the expectation is taken with respect to a random sample S drawn according to the lattice distribution $\mathcal{L}$ which is a concatenation of lattices $\mathcal{L}_1, \ldots, \mathcal{L}_m$ and where $\widehat{p}_S$ denotes the LM derived by training on sample S.

If we ignore computational considerations, the model defined in (3) can be constructed as an interpolation[3] of individual models $\widehat{p}_S$ with weights $\Pr_{S\sim\mathcal{L}}[S]$. In practice, it is not feasible to enumerate all possible samples S. However, suppose that there exists a function f such that for each n-gram $\mathbf{w}$, $\widehat{p}_S(\mathbf{w}) = f(c_S(\mathbf{w}))$. In other words, the estimate of the probability of $\mathbf{w}$ assigned by the model only depends on the count of that n-gram in the sample. Then it follows that

$$\mathop{\mathbb{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{w})] = \sum_{k=0}^{\infty} q_{\mathcal{L}}(k, \mathbf{w}) f(k), \tag{4}$$

where $q_{\mathcal{L}}(k, \mathbf{w}) = \sum_S \Pr_{S\sim\mathcal{L}}[c_S(\mathbf{w}) = k]$ is the probability that n-gram $\mathbf{w}$ occurs k times in the sample drawn according to $\mathcal{L}$.

In order to admit Eq. (4), we make some simplifying assumptions on the form of the underlying language models that determine $p_S(\mathbf{w})$ in Eq. (3). In particular, we assume they have the following variant form of a Katz language model:

$$\widehat{\Pr}_S(w|\mathbf{h}) = \begin{cases} \overline{d}_{c_S(\mathbf{h}w)} \frac{c_S(\mathbf{h}w)}{c_S(\mathbf{h})} & \text{if } c_S(\mathbf{h}w) > 0, \\ \beta_{\mathbf{h}}(\mathcal{L})\widehat{\Pr}_S(w|\mathbf{h}') & \text{otherwise,} \end{cases} \tag{5}$$

with $\mathbf{h}' <_{\text{suf}} \mathbf{h}$ and with the discount factors

$$\overline{d}_k = \frac{(k+1)\overline{n}_{k+1}}{k\overline{n}_k}, \tag{6}$$

where $\overline{n}_k = \sum_{\mathbf{w}} q_{\mathcal{L}}(k, \mathbf{w})$. Note that the normalization constant $\beta_{\mathbf{h}}(\mathcal{L})$ and discount factors $\overline{d}_k$s do not depend on a particular sample S and instead take into account the full information in $\mathcal{L}$.[4] This dependence on global information in $\mathcal{L}$ instead of a particular sample S is key to verifying Eq. (4). We describe the choice of $\beta_{\mathbf{h}}(\mathcal{L})$ below. The derivation of the discount factors $\overline{d}_k$s makes use of a generalized Good-Turing estimate as described in Section 3.5.

It follows that, for any n-gram $\mathbf{h}w$ observed in the lattice $\mathcal{L}$, the following holds:

$$\begin{aligned} \mathop{\mathrm{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{h}w)] &= q_{\mathcal{L}}(0, \mathbf{h}w)\beta_{\mathbf{h}}(\mathcal{L}) \mathop{\mathrm{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{h}'w)] \\ &\quad + \sum_{k=1}^{K} q_{\mathcal{L}}(k,\mathbf{h}w)\frac{\overline{d}_k k}{|S|} + \sum_{k=K+1}^{m} q_{\mathcal{L}}(k,\mathbf{h}w)\frac{k}{|S|} \\ &= q(0, \mathbf{h}w)\beta_{\mathbf{h}}(\mathcal{L}) \mathop{\mathrm{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{h}'w)] + \frac{\lambda_{\mathbf{h}w,\mathcal{L}}}{|S|} \\ &\quad + \sum_{k=1}^{K} q_{\mathcal{L}}(k, \mathbf{h}w)\frac{k}{|S|}\left(\overline{d}_k - 1\right), \end{aligned} \tag{7}$$

where $\lambda_{\mathbf{w},\mathcal{L}} = \sum_{k=0}^{m} q_{\mathcal{L}}(k, \mathbf{w})k$ is the expected count of $\mathbf{w}$ with respect to $\mathcal{L}$. Otherwise, if $\mathbf{h}w$ is not observed in $\mathcal{L}$ then $\mathop{\mathrm{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{h}w)] = \beta_{\mathbf{h}}(\mathcal{L}) \mathop{\mathrm{E}}_{S\sim\mathcal{L}}[\widehat{p}_S(\mathbf{h}'w)]$. Conditioning on the history $\mathbf{h}$ leads to the following ensemble model:

$$\widehat{\Pr}(w|\mathbf{h}) = \begin{cases} \frac{1}{\lambda_{\mathbf{h},\mathcal{L}}}\left[\lambda_{\mathbf{h}w,\mathcal{L}} + \sum_{k=1}^{K} q_{\mathcal{L}}(k, w)k\left(\overline{d}_k - 1\right)\right] \\ \quad + q(0, \mathbf{h}w)\beta_h(\mathcal{L})\widehat{\Pr}(w|\mathbf{h}') & \text{if } \mathbf{h}w \in \mathcal{L}, \\ \beta_{\mathbf{h}}(\mathcal{L})\widehat{\Pr}(w|\mathbf{h}'), & \text{otherwise,} \end{cases} \tag{8}$$

where $\mathbf{h}' <_{\text{suf}} \mathbf{h}$, and $\beta_{\mathbf{h}}(\mathcal{L})$ is a normalization constant. Remarkably, even though we started with back-off models not guaranteed to represent proper probability distributions due to

[3] This is the *Bayesian* rather than the linear interpolation of these models [2, 7].

[4] In *stupid back-off* [17] an even stronger assumption is made that $\beta_{\mathbf{h}}$ is the same constant for all histories.

the sample-independent back-off and discount factors, our final ensemble model does form a probability distribution. Note also that the only quantities that are required to construct this model are the $\lambda_{\cdot,\mathcal{L}}$s and the histograms $\{q(k,\cdot)\}_{k=1}^{K}$. In the next section, we present an efficient algorithm for computing these statistics.

3.4. Computing the Histograms $q(\cdot, \mathbf{w})$

For simplicity, assume that our sample consists of two lattices $\mathcal{T}$ and $\mathcal{U}$ such that $\Pr_{S\sim\mathcal{L}}[S] = \Pr_{T\sim\mathcal{T}}[T]\Pr_{U\sim\mathcal{U}}[U]$. We can compute the overall count probabilities for a sequence of words $\mathbf{w}$ from its components as follows:

$$\begin{aligned}
&q_{\mathcal{L}}(k,\mathbf{w})\\
&=\sum_{S}\Pr_{S\sim\mathcal{L}}[c_S(\mathbf{w})=k]\\
&=\sum_{T}\sum_{U}\sum_{j=0}^{k}\Pr_{T\sim\mathcal{T}}[c_T(\mathbf{w})=j]\Pr_{U\sim\mathcal{U}}[c_U(\mathbf{w})=k-j]\\
&=\sum_{j=0}^{k}\left(\sum_{T}\Pr_{T\sim\mathcal{T}}[c_T(\mathbf{w})=j]\right)\left(\sum_{U}\Pr_{U\sim\mathcal{U}}[c_U(\mathbf{w})=k-j]\right)\\
&=\sum_{j=0}^{k}q_{\mathcal{T}}(j,\mathbf{w})q_{\mathcal{U}}(k-j,\mathbf{w}). \qquad (9)
\end{aligned}$$

The expected count of a sequence word $\mathbf{w}$ becomes:

$$\lambda_{\mathbf{w},\mathcal{L}} = \lambda_{\mathbf{w},\mathcal{T}} + \lambda_{\mathbf{w},\mathcal{U}}. \qquad (10)$$

This computation can be straightforwardly extended to the general case of m lattices.

Finally, to further speed up the computation, we assume that an n-gram that is rare in the corpus occurs at most once on each lattice path. More precisely, we assume the following for each component distribution $\mathcal{T}$ and for each $k \leq K$:

$$q_{\mathcal{T}}(k,\mathbf{w}) = \begin{cases}\lambda_{\mathbf{w},\mathcal{T}} & \text{if } k=1\\ 1-\lambda_{\mathbf{w},\mathcal{T}} & \text{if } k=0.\end{cases} \qquad (11)$$

This assumption avoids computing the $q_{\mathcal{T}}$s directly for individual lattices and reduces the problem to computing $\lambda_{\mathbf{w}}$ for each $\mathcal{L}_i$ and then using (9), (10) and (11) to find the global statistics for $\mathcal{L}$.

3.5. Good-Turing Estimation

In this section, we provide a detailed derivation of the discount factors used in the certain and uncertain data cases based on Good-Turing estimation [13, 18, 19, 20, 21].

For the Katz back-off language models described in Section 2, the discount factors d_k's are completely specified by the following system of linear equations

$$\begin{aligned}
\sum_{c(\mathbf{w})=k} d_k\frac{c(\mathbf{w})}{m} &= M_k, \quad k\geq 1\\
\sum_{c(\mathbf{w})>0} d_{c(\mathbf{w})}\frac{c(\mathbf{w})}{m} &= 1-M_0, \qquad (12)
\end{aligned}$$

where $M_k = \Pr(\mathbf{w}\in S_k)$ is the probability mass of n-grams that occur precisely k times in the training corpus S and the left-hand side is an estimate of that quantity based on the model. Solving for d_k leads to $d_k = \frac{m}{n_k k}M_k$. This solution cannot be used directly since M_k is typically unknown. In practice, M_k is replaced by the Good-Turing estimator [18] denoted by G_k and defined to be

$$G_k = \frac{k+1}{m}n_{k+1}, \qquad (13)$$

where $n_k = |S_k|$, which yields the expression for d_k in (2). In the setting of uncertain data, we replace G_k with its expected value with respect to the lattice distribution $\mathcal{L}$:

$$\mathcal{G}_k = \mathop{\mathrm{E}}_{S\sim\mathcal{L}}[G_k] = \frac{k+1}{m}\overline{n}_{k+1}. \qquad (14)$$

More precisely, replacing M_k with G_k in (12) and taking the expectation with respect to $\mathcal{L}$ on both sides leads to the following system:

$$\begin{aligned}
\overline{d}_k\overline{n}_k\frac{k}{m} &= \mathcal{G}_k, \quad k\geq 1\\
\sum_{k=1}^{\infty}\overline{d}_k\overline{n}_k\frac{k}{m} &= 1-\mathcal{G}_0. \qquad (15)
\end{aligned}$$

Solving this system for $\overline{d}_k$ leads to the expression (6).

4. Experiments

We carried out experiments on videos sampled from Google Preferred channels [22] using multi-pass automatic speech recognition. Google Preferred is a program that allows advertisers access to the top 5% most popular channels with the most passionate audiences on YouTube. Google Preferred channels are determined algorithmically based on popularity and passion metrics including watch time, likes, shares, and fanships to surface among the top 5% of channels. Those channels are packaged into lineups that brands can buy to align with engaged audiences and scarce content on YouTube. A lineup corresponds to a category of video. For this test set, we selected a subset of these videos from Preferred channels: we picked recent videos (after 1/1/2012) with moderate video length (120-600secs), and high video view count. For this paper, we have 13 Preferred lineups, including: Anime & Teen Animation, Parenting & Children Interest, Science & Education, and Sports, among others. We used one lineup, Video Games, as a development set, to explore meta-parameters and best practices for model adaptation.

The baseline acoustic model is comprised of 2 LSTM layers, where each LSTM layer admits 800 cells, a 512-unit projection layer for dimensionality reduction [23], and a softmax layer with 6398 context-dependent triphone states [24] clustered using decision trees [25]. We use a reduced X-SAMPA phonetic alphabet of 40 phones plus silence. The features are 40-dimensional log mel-spaced filterbank coefficients, without any temporal context. The acoustic training data set is 764 hours of transcribed video data as described in [9]. The base language model is a Katz smoothed 5-gram model with 30M n-grams and a vocabulary of approximately 2M words.

To adapt the language model for a given lineup, we train a trigram language model on the ASR 1-best or lattice output of the baseline system, pruned to include a maximum of 30,000 n-grams.[5] We then combine this model with the baseline 5-gram language model using simple linear interpolation with an experimentally selected mixing parameter.

[5] We prune to this number of n-grams to somewhat control for different numbers of n-grams being used to adapt the model, depending on whether one-best transcripts are used or full lattice counts, as well as in conditions with varying amounts of count thresholding of n-grams. Given the amount of adaptation data, we never found any benefit from n-grams of higher order than trigram.

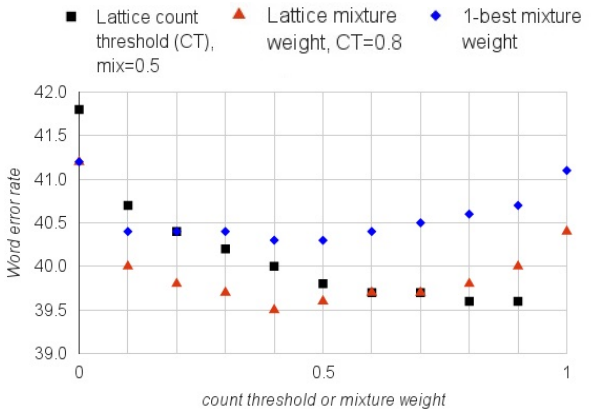

Figure 1: Parameter sweep on the dev set of: (1) count threshold values for fractional counts from lattices, with mixture weight fixed at 0.5; and (2) weights for mixing with the baseline model for both lattice and one-best methods, with fractional count thresholding fixed at 0.8.

For using the one-best transcripts, we experimented with three methods with the dev set: thresholding on posterior probability to include only those transcripts with sufficient confidence; using the posterior probability to weight the counts derived from the one-best transcripts[6]; and using all one-best transcripts for the entire lineup with no weighting. While confidence-based thresholding or weighting have been used successfully for tasks such as voice search [10], we found that neither improved upon simply using all unweighted one-best transcripts, most likely due to the relatively long utterances in the collection and the fact that confidence thresholds (or count weighting) favored the output for shorter utterances. For this reason, the one-best trials reported used no thresholding or count weighting to derive the models.

Our two main meta-parameters for the approach are count thresholding (based on expected frequency) for the lattice-based approach and the mixing weight $\alpha \in [0, 1]$ for the adaptation model[7]. Figure 1 shows a sweep over count thresholds for lattice-based counts (with the mixing parameter fixed at α=0.5); and mixing parameters for both lattice-based and one-best models, with count thresholding for the lattice-based model fixed at 0.8. Based on these, the mixture weight for adaptation was set to α=0.4 for both lattice-based and one-best adaptation models (i.e., 0.4 weight to the adaptation model, 0.6 weight to the baseline), and the count thresholding was set to 0.8, i.e., any n-gram with expected frequency below 0.8 was discarded.

Table 1 presents results on the dev lineup (Video Games) and the other (test) lineups, at the optimized meta parameters of 0.8 count thresholding and 0.4 mixture weight for adaptation. In some cases, the one-best trained adaptation models actually hurt performance relative to the baseline model; but this is never the case for lattice trained models, which generally provide larger improvements than one-best trained models, particularly when more adaptation data is available. Figure 2 plots the reduction in WER from the baseline system versus the size of the test set for both one-best and lattice-based adaptation. While the size of the test set does not explain all of the variance, there is a definite trend favoring larger test sets.

We also built Monte-Carlo sampled models of the sort described in Section 3.2 and used in [6], where k corpora were sampled for each lineup, and an adaptation language model was built by uniformly merging all k models, before mixing with the baseline language model. On the dev set, sampling and merging 100 models yielded performance only marginally better than just using one-best, and sampling and merging 1000 models just 0.1 better than that. This is still over a half percent absolute worse than our lattice-based fractional count methods. This may also be the result of having relatively long utterances, so that many more samples would be required to be competitive with our approach. Of course, even if the accuracies of such sampling based methods were commensurate, our lattice-based approach provides the more efficient and scalable solution.

[6]This is essentially our fractional count method, but applied to only a single path for each utterance.

[7]The model is mixed using α times the adaptation model probability plus $1 - \alpha$ times the baseline model probability.

		Word Error Rate (WER)			
	Tokens	Base-	Adapted		
Google Preferred Lineup	×1000	line	1-best	Lattice	Δ
Video Games (dev set)	23.8	41.2	40.3	39.5	1.6
Anime & Teen Animation	4.3	29.9	30.3	29.7	0.2
Beauty & Fashion	37.3	29.6	29.1	28.0	1.6
Cars, Trucks & Racing	5.7	21.6	22.1	21.6	0.0
Comedy	9.3	55.3	54.9	54.9	0.4
Entertainment & Pop Culture	27.8	39.3	39.4	38.9	0.4
Food & Recipes	11.4	42.6	43.0	41.7	0.9
News	12.3	27.4	27.3	26.7	0.7
Parenting & Children Interest	11.7	38.0	38.4	37.1	0.9
Science & Education	15.9	22.0	22.4	21.5	0.5
Sports	6.7	47.3	47.9	47.3	0.0
Technology	23.7	23.1	23.1	22.3	0.8
Workouts, Weightlifting & Wellness	13.2	31.0	30.5	29.1	1.8
All test lineups	179.2	28.8	28.8	28.0	0.8

Table 1: Performance on dev set (row 1) and test channel lineups with count thresholding at 0.8 and model mixing at 0.4.

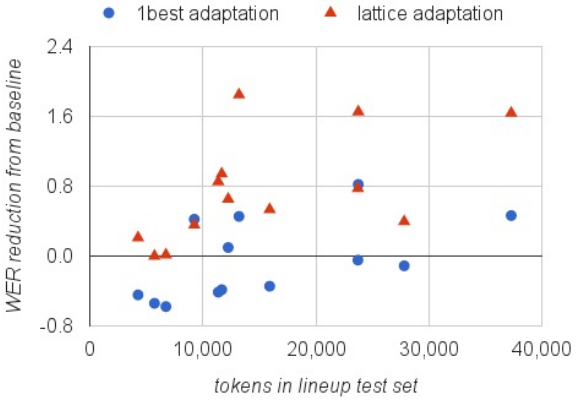

Figure 2: WER reduction vs. number of tokens in the test set, for both one-best and lattice-based adaptation for each of the lineups examined.

5. Conclusion

We presented a new algorithm for learning generalized Katz back-off models with fractional counts derived from the uncertain output of ASR systems. Semi-supervised adaptation using full lattices with our generalized Katz back-off algorithm yielded a 0.8 absolute WER reduction in aggregate vs. the baseline, and did not hurt performance in any of the individual lineups. In contrast, adapting on one-best output did hurt performance in many lineups, resulting in no overall WER reduction.

6. Acknowledgments

The work of M. Mohri and V. Kuznetsov was partly funded by NSF IIS-1117591 and CCF-1535987.

7. References

[1] B. Roark, R. Sproat, C. Allauzen, M. Riley, J. Sorensen, and T. Tai, "The OpenGrm open-source finite-state grammar software libraries," in *Proceedings of the ACL 2012 System Demonstrations*, 2012, pp. 61–66. [Online]. Available: http://ngram.opengrm.org/

[2] A. Stolcke, "Error modeling and unsupervised language modeling," in *Proceedings of the 2001 NIST Large Vocabulary Conversational Speech Recognition Workshop*, Linthicum, Maryland, May 2001.

[3] R. Gretter and G. Riccardi, "On-line learning of language models with word error probability distributions," in *Proceedings of ICASSP*, 2001, pp. 557–560.

[4] M. Bacchiani and B. Roark, "Unsupervised language model adaptation," in *Proceedings of ICASSP*, 2003, pp. 224–227.

[5] A. Park, T. J. Hazen, and J. R. Glass, "Automatic processing of audio lectures for information retrieval: Vocabulary selection and language modeling." in *Proceedings of ICASSP*, 2005, pp. 497–500.

[6] M. Bacchiani, M. Riley, B. Roark, and R. Sproat, "MAP adaptation of stochastic grammars," *Computer Speech & Language*, vol. 20, no. 1, pp. 41–68, 2006.

[7] C. Allauzen and M. Riley, "Bayesian language model interpolation for mobile speech input," in *Proceeding of Interspeech*, 2011, pp. 1429–1432.

[8] E. Leeuwis, M. Federico, and M. Cettolo, "Language modeling and transcription of the TED corpus lectures," in *Proceedings of ICASSP*, vol. I, 2003, pp. 232–235.

[9] H. Liao, E. McDermott, and A. Senior, "Large scale deep neural network acoustic modeling with semi-supervised training data for YouTube video transcription," in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding (ASRU)*, 2013, pp. 368–373.

[10] F. Beaufays and B. Strope, "Language model capitalization," in *Proceedings of ICASSP*, 2013, pp. 6749–6752.

[11] M. Mohri, "Learning from uncertain data," in *Proceedings of COLT*, 2003, pp. 656–670.

[12] C. Allauzen, M. Mohri, and B. Roark, "Generalized algorithms for constructing language models," in *Proceedings of ACL*, 2003, pp. 40–47.

[13] S. M. Katz, "Estimation of probabilities from sparse data for the language model component of a speech recognizer," *IEEE Transactions on Acoustics, Speech and Signal Processing*, vol. 35, no. 3, pp. 400–401, Mar 1987.

[14] R. Kneser and H. Ney, "Improved backing-off for m-gram language modeling," in *Proceedings of ICASSP*, 1995, pp. 181–184.

[15] C. Chelba, T. Brants, W. Neveitt, and P. Xu, "Study on interaction between entropy pruning and Kneser-Ney smoothing," in *Proceedings of Interspeech*, 2010, p. 24222425.

[16] C. Chelba, J. Schalkwyk, T. Brants, V. Ha, B. Harb, W. Neveitt, C. Parada, and P. Xu, "Query language modeling for voice search," in *Proceedings of the IEEE Workshop on Spoken Language Technology*, 2010, pp. 127–132.

[17] T. Brants, A. C. Popat, P. Xu, F. J. Och, and J. Dean, "Large language models in machine translation," in *Proceedings of EMNLP-CoNLL*, 2007, pp. 858–867.

[18] I. J. Good, "The population frequencies of species and the estimation of population," *Biometrika*, pp. 237–264, 1953.

[19] D. A. McAllester and R. E. Schapire, "On the convergence rate of Good-Turing estimators," in *Proceedings of COLT*, 2000, pp. 1–6.

[20] E. Drukh and Y. Mansour, "Concentration bounds for unigrams language model," in *Proceedings of COLT*, 2004, pp. 170–185.

[21] A. Orlitsky and A. T. Suresh, "Competitive distribution estimation: Why is Good-Turing good," in *Advances in NIPS*, 2015, pp. 2143–2151.

[22] "Google Preferred Lineup Explorer - YouTube," Mar. 2016. [Online]. Available: http://youtube.com/yt/lineups/

[23] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of Interspeech*, 2014.

[24] L. Bahl, P. de Souza, P. Gopalkrishnan, D. Nahamoo, and M. Picheny, "Context dependent modelling of phones in continuous speech using decision trees," in *Proc. DARPA Speech and Natural Language Processing Workshop*, 1991, pp. 264–270.

[25] S. Young, J. Odell, and P. Woodland, "Tree-based state tying for high accuracy acoustic modelling," in *Proc. ARPA Workshop on Human Language Technology*, 1994, pp. 307–312.