WHOLE-SENTENCE EXPONENTIAL LANGUAGE MODELS: A VEHICLE FOR LINGUISTIC-STATISTICAL INTEGRATION

Ronald Rosenfeld, Stanley F. Chen[†] *and Xiaojin Zhu*

School of Computer Science
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213
{roni,sfc,zhuxj}@cs.cmu.edu

ABSTRACT

We introduce an exponential language model which models a whole sentence or utterance as a single unit. By avoiding the chain rule, the model treats each sentence as a "bag of features", where features are arbitrary computable properties of the sentence. The new model is computationally more efficient, and more naturally suited to modeling global sentential phenomena, than the conditional exponential (e.g. Maximum Entropy) models proposed to date. Using the model is straightforward. Training the model requires sampling from an exponential distribution. We describe the challenge of applying Monte Carlo Markov Chain (MCMC) and other sampling techniques to natural language, and discuss smoothing and step-size selection. We then present a novel procedure for feature selection, which exploits discrepancies between the existing model and the training corpus. We demonstrate our ideas by constructing and analyzing competitive models in the Switchboard domain, incorporating lexical and syntactic information.

1. MOTIVATION AND OUTLINE

Conventional statistical language models estimate the probability of a sentence s by using the chain rule to decompose it into a product of conditional probabilities:

$$\begin{aligned}\Pr(s) &\stackrel{\text{def}}{=} \Pr(w_1 \ldots w_n) \\ &= \prod_{i=1}^{n} \Pr(w_i | w_1 \ldots w_{i-1}) \\ &\stackrel{\text{def}}{=} \prod_{i=1}^{n} \Pr(w_i | h_i)\end{aligned}$$

where $h_i \stackrel{\text{def}}{=} \{w_1, \ldots, w_{i-1}\}$ is the *history* when predicting word w_i. The vast majority of work in statistical language modeling is devoted to estimating terms of the form $\Pr(w|h)$.

The application of the chain rule is technically harmless since it uses an exact equality, not an approximation. This practice is also understandable from a historical perspective (statistical language modeling grew out of the statistical approach to speech recognition, where the search paradigm requires estimating the probability of individual words). Nonetheless, it is not always desirable. Terms like $\Pr(w|h)$ may not be the best way to think about estimating $\Pr(s)$:

1. Global sentence information such as grammaticality or semantic coherence is awkward to encode in a conditional framework. Some grammatical structure was captured in the structured language model of [1] and in the conditional exponential model of [2]. But such structure had to be formulated in terms of partial parse trees and left-to-right parse states. Similarly, modeling of semantic coherence was attempted in the conditional exponential model of [3], but had to be restricted to a limited number of pairwise word correlations.

2. External influences on the sentence (for example, the effect of preceding utterances, or dialog level variables) are equally hard to encode efficiently. Furthermore, such influences must be factored into the prediction of every word in the current sentence, causing small but systematic biases in the estimation to be compounded.

3. $\Pr(w|h)$ is typically approximated by $\Pr(w_i|w_{i-k+1}, \ldots, w_{i-1})$ for some small k (the Markov assumption). Even if such a model is improved by including longer distance information, it still makes many implicit independence assumptions. It is clear from looking at language data that these assumptions are often patently false, and that there are significant global dependencies both within and across sentences.

[†] Currently with IBM T.J. Watson Research Center, Yorktown Heights, NY 10598. e-mail: stanchen@watson.ibm.com.

As a simple example of the limitations of the chain rule approach, consider one aspect of a sentence: its length. In an n-gram based model, the effect of the number of words in the utterance on its probability cannot be modeled directly. Rather, it is an implicit consequence of the n-gram prediction. This is later corrected during speech recognition by a "word insertion penalty," the usefulness of which proves that length is an important feature. However, the word insertion penalty can only model length as a geometric distribution, which does not fit well with empirical data, especially for short utterances.

As an alternative to the conventional conditional formulation, this paper proposes a new exponential language model which directly models the probability of an entire sentence or utterance. The new model is conceptually simpler, and more naturally suited to modeling whole-sentence phenomena, than the conditional exponential models proposed earlier. By avoiding the chain rule, the model treats each sentence or utterance as a "bag of features", where features are arbitrary computable properties of the sentence. The single, universal normalizing constant cannot be computed exactly, but this does not interfere with training (done via sampling) or with use. Using the model is computationally straightforward. Training the model depends crucially on efficient sampling of sentences from an exponential distribution.

In what follows, Section 2 introduces the model and contrasts it with the conditional exponential models proposed to date. Section 3 discusses training the model: it lists several techniques for sampling from exponential distributions, shows how to apply them to the domain of natural language sentences, and compares their relative efficacies. Step-size selection and smoothing are also discussed here. Section 4 describes experiments we performed with this model, incorporating lexical and syntactic information. Section 5 analyzes the results of the experiments, and Section 6 summarizes and discusses our ongoing effort and future directions. Various portions of this work were first described in [4, 5, 6].

2. WHOLE SENTENCE EXPONENTIAL MODELS

A whole sentence exponential language model has the form:

$$p(s) = \frac{1}{Z} \cdot p_0(s) \cdot \exp(\sum_i \lambda_i f_i(s)) \qquad (1)$$

where the $\{\lambda_i\}$'s are the parameters of the model, Z is a universal normalization constant which depends only on the $\{\lambda_i\}$'s, and the $\{f_i(s)\}$'s are arbitrary computable properties, or *features*, of the sentence s. $p_0(s)$ is any arbitrary initial distribution, sometimes loosely referred to as the "prior". For example, $p_0(s)$ might be the uniform distribution, or else it might be derived (using the chain rule) from a conditional distribution such as an n-gram.

The features $\{f_i(s)\}$ are selected by the modeler to capture those aspects of the data they consider appropriate or profitable. These can include conventional n-grams, longer-distance dependencies, global sentence properties, as well as more complex functions based on part-of-speech tagging, parsing, or other types of linguistic processing.

2.1. Using the Whole Sentence Model

To use the whole sentence exponential model to estimate the probability of a given sentence s, one need only calculate $p_0(s)$ and the values of the various features $f_i(s)$, and use Equation 1. Thus using the model is straightforward and (as long as the features are not too complex) computationally trivial. Because the features could depend on any part of the sentence, they can in general only be computed after the entire sentence is known. Therefore, when used for speech recognition, the model is not suitable for the first pass of the recognizer, and should instead be used to re-score N-best lists.

2.2. Whole Sentence Maximum Entropy Models

The term "exponential model" refers to any model of the form (1). A particular type of such a model is the so-called "Maximum Entropy" (ME) model, where the parameters are chosen so that the distribution satisfies certain linear constraints. Specifically, for each feature $f_i(s)$, its expectation under $p(s)$ is constrained to a specific value K_i:

$$E_p f_i = K_i \quad . \qquad (2)$$

These target values are typically set to the expectation of that feature under the empirical distribution $\tilde{p}$ of some training corpus $\{s_1, \ldots, s_N\}$ (for binary features, this simply means their frequency in the corpus). Then, the constraint becomes:

$$\sum_s p(s) \cdot f_i(s) = E_{\tilde{p}} f_i = \frac{1}{N} \sum_{j=1}^{N} f_i(s_j) \quad . \qquad (3)$$

If the constraints (2) are consistent, there exists a unique solution within the exponential family (1) which satisfies them. Among all (not necessarily exponential) solutions to equations (2), the exponential solution is the one closest to the initial distribution $p_0(s)$ in the Kullback-Liebler sense, and is thus called the Minimum Divergence or Minimum Discrimination Information (MDI) solution. If $p_0(s)$ is uniform, this becomes simply the Maximum Entropy (ME) solution[1]. Furthermore, if the feature target values K_i are

[1] In the literature, the term "Maximum Entropy" or ME is used loosely to refer to both situations, i.e. regardless of whether the initial distribution p_0 is uniform. We will follow this practice here.

the empirical expectations over some training corpus (as in equations (3)), the MDI or ME solution is also the Maximum Likelihood solution of the exponential family. For more information, see [7, 8, 3].

The MDI or ME solution can be found by an iterative procedure such as the Generalized Iterative Scaling (GIS) algorithm [9]. GIS starts with arbitrary λ_i's. At each iteration, the algorithm improves the $\{\lambda_i\}$ values by comparing the expectation of each feature under the current p to the target value, and modifying the associated λ. In particular, we take

$$\lambda_i \leftarrow \lambda_i + \frac{1}{F_i} \log \frac{K_i}{E_p[f_i]} \tag{4}$$

where F_i determines the step size (see Section 3.2).

In training a whole-sentence Maximum Entropy model, computing the expectations $E_p[f_i] = \sum_s p(s) \cdot f_i(s)$ requires a summation over all possible sentences s, a clearly infeasible task. Instead, we estimate $E_p[f_i]$ by sampling from the distribution $p(s)$ and using the sample expectation of f_i. Sampling from an exponential distribution is a non-trivial task. Efficient sampling is crucial to successful training. Sampling techniques will be discussed in detail in Section 3.1. As will be shown later on in this paper, with these techniques it is possible to train whole-sentence ME models using very large corpora and very many features.

As mentioned earlier, the term "exponential models" refers to all models of the form (1), whereas the term "ME models" refers to exponential models where the parameters are set to satisfy equation (2). Most of this paper deals with ME models. For a different training criterion for exponential models, see Section 2.5.

2.3. Comparison to Conditional ME Models

It is instructive to compare the whole-sentence ME model with conditional ME models, which have seen considerable success recently in language modeling [10, 11, 8, 3]. The conditional model usually has the form:

$$p(w|h) = \frac{1}{Z(h)} \cdot p_0(w|h) \cdot \exp\left[\sum_i \lambda_i f_i(h,w)\right] \tag{5}$$

where the features are functions of a specific word-history pair, and so is the baseline p_0. More importantly, Z here is not a true constant — it depends on h and thus must be recomputed during training for each word position in the training data. Namely, for each training datapoint (h, w), one must compute

$$Z(h) \stackrel{\text{def}}{=} \sum_{w \in \mathcal{V}} p_0(w|h) \exp[\sum_i \lambda_i f_i(h,w)]$$

where $\mathcal{V}$ is the vocabulary. This computational burden is quite severe: training a model that incorporates n-gram and long-distance word correlation features on some 40 million words can take hundreds of days of computation [3, p.210]. It is this bottleneck that has hindered the widespread use of the ME framework in language modeling. In comparison, in our new model the universal normalization constant need not be calculated (see Section 2.4). This not only speeds up training, but it also significantly speeds up using the model in applications such as speech recognition. In fact, when used with simple features, our model can be applied with roughly the same amount of computation as an n-gram model employing a comparable number of parameters.

The main drawbacks of the conditional ME model are thus the severe computational bottleneck of training (especially of computing $Z(h)$), and the difficulties in modeling whole-sentence phenomena. The whole-sentence ME model addresses both of these issues.

A final note of comparison: A whole-sentence ME model incorporating the same features as a conditional ME model is in fact not identical to the latter. This is because the training procedure used for conditional ME models restricts the computation of the feature expectations to histories observed in the training data (see [8] or [3, section 4.4]). This biases the solution in an interesting and sometimes appropriate way. For example, consider word trigger features of the form

$$f_{X \to Y}(h,w) = \begin{cases} 1 & \text{if } X \in h, w = Y \\ 0 & \text{otherwise} \end{cases}$$

and specifically consider the two features $A \longrightarrow Z$ and $B \longrightarrow Z$ for some words A, B, and Z. If A and B are correlated in the training data, this will affect the solution of the conditional ME model. In fact, if they are perfectly correlated, always co-occurring in the same sentence, the resulting λ's will likely be one half of what their value would have been if only one of the features were used. This is beneficial to the model, since it captures correlations that are likely to recur in new data. However, a whole-sentence ME model incorporating the same features will not use the correlation between A and B, unless it is explicitly instructed to do so via a separate feature. This is because the training data is not actually used in whole-sentence ME training, except initially to derive the features' target values.

2.4. Normalization and Perplexity

Just as it is infeasible to calculate exactly feature expectations for whole-sentence models, it is equally infeasible to compute the normalization constant $Z = \sum_s p_0(s) \cdot \exp(\sum_i \lambda_i f_i(s))$. Fortunately, this is not necessary for training: sampling (and hence expectation estimation) can be done without knowing Z, as will be shown in Section 3. Using the model as part of a classifier (*e.g.*, a speech recognizer) does not require knowledge of Z either, because the relative ranking of the different classes is not changed by a

single, universal constant. Notice that this is not the case for conditional exponential models.

Nonetheless, at times it may be desirable to approximate Z, perhaps in order to compute perplexity. With the whole-sentence model this can be done as follows.

Let $R(s) = \exp(\sum_i \lambda_i f_i(s))$ be the unnormalized modification made to the initial model $p_0(s)$. Then $p(s) = \frac{1}{Z} p_0(s) R(s)$. By the normalization constraint we have:

$$\sum_s p(s) = \frac{1}{Z} \sum_s p_0(s) \cdot R(s) = \frac{1}{Z} E_{p_0}[R(S)] = 1$$

From which we get:

$$Z = E_{p_0}[R(S)].$$

Thus Z can be approximated to any desired accuracy from a suitably large sample T_0 of sentences drawn from p_0. Often, p_0 is based on an n-gram. A reasonably efficient sampling technique for n-grams is described later.

To estimate reduction in per-word perplexity (PP) over the p_0 baseline, let T_e be a test set, and let $\#S_e$ and $\#W_e$ be the number of sentences and words in it, respectively. By definition

$$PP(T_e) = p(T_e)^{-\frac{1}{\#W_e}}$$

It follows then that the perplexity reduction ratio is:

$$\frac{PP_{\text{EXP}}(T_e)}{PP_0(T_e)} = \left(\frac{Z}{\sqrt[\#S_e]{\prod_{s \in T_e} R(s)}} \right)^{\frac{\#S_e}{\#W_e}}$$

Substituting in the estimation of Z, the estimated perplexity reduction is:

$$\frac{PP_{\text{EXP}}(T_e)}{PP_0(T_e)} \approx \left(\frac{\text{arithmetic mean}[R(s)]}{\text{geometric mean}[R(s)]} \right)^{\frac{\#S_e}{\#W_e}}$$

where the arithmetic mean is taken with respect to p_0 and the geometric mean with respect to T_e. Interestingly, if $T_e = T_0$, i.e. if the test set is also sampled from the baseline distribution, it follows from the law of inequality of averages that the new perplexity will always be higher. This, however, is appropriate because any "correction" to an initial probability distribution will assign a lower likelihood (and hence higher perplexity) to data generated by that distribution.

2.5. Discriminative Training

Until now we discussed primarily Maximum Entropy or Maximum Likelihood training. However, whole-sentence exponential models can also be trained to directly minimize the error rate in an application such as speech recognition. This is known as Minimum Classification Error (MCE) training, or Discriminative Training. The log-likelihood of a whole-sentence exponential model with k features is given by

$$\log p(s) = -\log Z + \log p_0(s) + \sum_{i=1}^{k} [\lambda_i f_i(s)]$$

The last term is a weighted sum, which can be directly (albeit only locally) optimized for MCE using a heuristic grid search such as Powell's algorithm, which searches the space defined by the λ's. In fact, the second term ($\log p_0(s)$) can also be assigned a weight, and scores not related to language modeling can be added to the mix as well, for joint optimization. This is a generalization of the "language weight" and "word insertion penalty" parameters currently used in speech recognition. For an attempt in this direction, see [12].

2.6. Conditional Whole-Sentence Models

So far we have discussed non-conditional models of the form $p(s)$. In order to model cross-sentence effects, one can re-introduce the conditional form of the exponential model, albeit with some modifications. Let $\hat{h}$ refer to the *sentence history*, namely the sequence of sentences from the beginning of the document or conversation up to (but not including) the current sentence. The model then becomes:

$$p(s|\hat{h}) = \frac{1}{Z(\hat{h})} \cdot p_0(s|\hat{h}) \cdot \exp(\sum_i \lambda_i f_i(\hat{h}, s))$$

Although the normalization constant is no longer universal, it is still not needed for N-best rescoring of a speech recognizer's output. This is because rescoring is typically done one sentence at a time, with all competing hypotheses sharing the same sentence history.

We will not pursue models of this type any further in this paper, except to note that they can be used to exploit session wide information, such as topics and other dialog level features.

3. MAXIMUM ENTROPY MODEL TRAINING

3.1. Sampling

Since explicit summation over all sentences s is infeasible, we will estimate the expectations $E_p[f_i]$ by sampling. In this section, we describe several statistical techniques for sampling from exponential distributions, and evaluate their efficacy for generating natural language sentences.

Gibbs Sampling [13] is a well known technique for sampling from exponential distributions. It was used in [14] to sample from the population of character strings. We will now describe how to use it to generate whole sentences from

an unnormalized joint exponential distribution, then present alternative methods which are more efficient in this domain.

To generate a single sentence from $p(s)$, start from any arbitrary sentence s, and iterate as follows:

1. Choose a word position i (either randomly or by cycling through all word positions in some order).

2. Let s_w^i be the sentence produced by replacing the word in position i in sentence s with the word w. For each word w in the vocabulary $\mathcal{V}$, calculate $p(s_w^i) = p_0(s_w^i) \exp(\sum_i \lambda_i f_i(s_w^i))$.

3. Choose a word at random according to the distribution $\{p(s_w^i)\}_{w \in \mathcal{V}}$. Place that word in position i in the sentence. This constitutes a single step in the random walk in the underlying Markov field.

To allow transitions into sentences of any length, we do the following:

- The end-of-sentence position is also considered for replacement by an ordinary word, which effectively lengthens the sentence by one word.

- When the last word position in the sentence is picked, the end-of-sentence symbol `</s>` is also considered. If chosen, this effectively shortens the sentence by one word.

After enough iterations of the above procedure, the resulting sentence is guaranteed to be an unbiased sample from the Gibbs distribution $p(s)$.[2]

Generating sample sentences from a Gibbs distribution as described above is quite slow. To speed things up, the following variations are useful:

- Draw the initial sentence s from a "reasonable" distribution, such as a unigram based on the training data, or from p_0. This tends to reduce the necessary number of iterations per step.

- For an initial sentence s use the final (or some intermediate) sentence from a previous random walk. This again tends to reduce the necessary number of iterations. However, the resulting sentence may be somewhat correlated with the previous sample[3].

- At each iteration, consider only a subset of the vocabulary for replacement. Any subset can be chosen as long as the underlying Markov Chain remains ergodic. This trades off the computational cost per iteration against the mixing rate of the Markov chain (that is, the rate at which the random walk converges to the underlying equilibrium distribution).

Even with these improvements, Gibbs sampling is not the most efficient for this domain, as the probability of a great many sentences must be computed to generate each sample. **Metropolis sampling** [15], another Markov Chain Monte Carlo technique, appears more appropriate for this situation. An initial sentence is chosen as before. For each chosen word position i, a new word v is proposed from a distribution $q(v)$ to replace the original word w in that position, resulting in a proposed new sentence s_v^i. This new sentence is *accepted* with probability

$$\min[1, \frac{p(s_v^i) \cdot q(w)}{p(s_w^i) \cdot q(v)}] \quad .$$

Otherwise, the original word w is retained. After all word positions have been examined, the resulting sentence is added to the sample, and this process is repeated.[4] The distribution $q(v)$ used to generate new word candidates for each position affects the sampling efficiency; in the experiments reported in this paper, we used a unigram distribution.

As in Gibbs sampling, adapting the Metropolis algorithm to sentences of variable-length requires care. In one solution, we pad each sentence with end-of-sentence tokens `</s>` up to a fixed length l. A sentence becomes shorter if the last non-`</s>` token is changed to `</s>`, longer if the first `</s>` token is changed to something else.

In applying Metropolis sampling, instead of replacing a single word at a time it is possible to replace larger units. In particular, in **independence sampling** we consider replacing the whole sentence in each iteration. For efficiency, the distribution $q(s)$ used to generate new sentence candidates must be similar to the distribution $p(s)$ we are attempting to sample from.

In **importance sampling**, a sample $\{s_1, \ldots, s_M\}$ is generated according to some sentence distribution $q(s)$, which similarly must be close to $p(s)$ for efficient sampling. To correct the bias introduced by sampling from $q(s)$ instead of from $p(s)$, each sample s_j is counted $\frac{p(s_j)}{q(s_j)}$ times, so that we have

$$E_p[f_i] \approx \frac{\sum_{j=1}^{M} \frac{p(s_j)}{q(s_j)} f_i(s_j)}{\sum_{j=1}^{M} \frac{p(s_j)}{q(s_j)}} \quad . \tag{6}$$

Which sampling method is best depends on the nature of $p(s)$ and $q(s)$. We evaluated these methods (except Gibbs sampling, which proved too slow) on some of the models to

[2] It is not theoretically known how many iterations are needed to practically guarantee unbiased samples. In our experiments we used several thousands.

[3] This is known as the "long chain" approach. There is an ongoing and heated debate in the computational statistics community between "long chain" and "short chain" supporters.

[4] The sampling procedure is still correct if the current sentence is added to the sample after *each* word position is examined; however, this process becomes less well-defined when we consider variable-length sentences.

	sampling algorithm		
	Metropolis	independence	importance
$f_{1,4}$	0.38±0.07	0.438±0.001	0.439±0.001
$f_{5,8}$	0.10±0.02	0.1001±0.0004	0.1001±0.0006
$f_{9,12}$	0.08±0.01	0.0834±0.0006	0.0831±0.0006
$f_{13,16}$	0.073±0.008	0.0672±0.0005	0.0676±0.0007
$f_{17,\infty}$	0.37±0.09	0.311±0.001	0.310±0.002

Table 1: Mean and standard deviation (of mean) of feature expectation estimates for sentence-length features for three sampling algorithms over ten runs

be described in Section 4. These models employ a trigram as the initial distribution $p_0(s)$. (Generating sentences from a n-gram model can be done quite efficiently: one starts from the beginning-of-sentence symbol, and iteratively generates a single word according to the n-gram model and the specific context, until the end-of-sentence symbol is generated. Generating a single word from an n-gram model requires $o(|\mathcal{V}|)$ steps. While this computation is not trivial, it is far more efficient than sampling directly from an exponential distribution.) Therefore, taking $q(s)$ to be a trigram model for independence and importance sampling is very effective. To measure the effectiveness of the different sampling algorithms, we did the following. Using an exponential model with a baseline trigram trained on 3 million words of Switchboard text ([16]) and a vocabulary of some 15,000 words, for each of the sampling methods we generated ten independent samples of 100,000 sentences. We estimated the expectations of a set of features on each sample, and calculated the empirical variance in the estimate of these expectations over the ten samples. More efficient sampling algorithms should yield lower variances.

In our experiments, we found that independence sampling and importance sampling both yielded excellent performance, while word-based Metropolis sampling performed substantially worse. As an example, we estimated expectations for sentence-length features of the form

$$f_{l_1,l_2}(s) = \begin{cases} 1 & \text{if } l_1 \leq \text{length}(s) \leq l_2 \\ 0 & \text{otherwise} \end{cases}$$

over ten samples of size 100,000. In Table 1, we display the means and standard deviations of the ten expectation estimates for each of the five sentence-length features under three sampling algorithms.

The efficiency of importance and independence sampling depends on the distance between the generating distribution $q(s)$ and the desired distribution $p(s)$. If $q(s) = p_0(s)$, that distance will grow with each training iteration. Once the distance becomes too large, Metropolis sampling can be used for one iteration, say iteration k, and the resulting sample retained. Subsequent iterations can re-use that sample via importance or independence sampling with $q(s) = p^{[k]}(s)$. Note that, even if training were to stop at iteration k, $p^{[k]}$ is arguably a better model than the initial model p_0, since it has moved considerably (by our assumption) towards satisfying the feature constraints.

Using the techniques we discuss above, training a whole-sentence ME model is feasible even with large corpora and many features. And yet, training time is not negligible. Some ideas for reducing the computational burden which we have not yet explored include:

- Use only rough estimation (i.e. small sample size) in the first few iterations (we only need to know the direction and rough magnitude of the correction to the λ's); increase sample size when approaching convergence.
- Determine the sample size dynamically, based on the number of times each feature was observed so far.
- Add features gradually (this has already proven itself effective at least anecdotally, as reported in Section 4.1).

3.2. Step Size

In GIS, the step size for feature update is inversely related to the number of active features. As sentences typically have many features, this may result in very slow convergence. Improved Iterative Scaling (IIS) [14] uses a larger effective step size than GIS, but requires a great deal more bookkeeping.

However, when feature expectations are near their target value, IIS can be closely approximated with equation (4) where F_i is taken to be a weighted average of the feature sum over all sentences; *i.e.*, if the set of sentences s were finite, we would take

$$F_i = \frac{1}{\sum_s p(s) f_i(s)} \sum_s p(s) f_i(s) \sum_{i'} f_{i'}(s) \quad . \qquad (7)$$

In our implementation, we approximated F_i by summing only over the sentences in the sample used to calculate expectations. This technique resulted in convergence in all of our experiments.

3.3. Smoothing

From equation (4) we can see that if $E_{\tilde{p}}[f_i] = 0$ then we will have $\lambda_i \to -\infty$. To *smooth* our model, we use the fuzzy maximum entropy method first described by [17]: We introduce a Gaussian prior on λ_i values and search for the maximum *a posterior* model instead of the maximum likelihood model. This has the effect of changing Equation (2)

to

$$E_p f_i = K_i - \frac{\lambda_i}{\sigma_i^2}$$

for some suitable variance parameter σ_i^2. With this technique, we found that over-training (overfitting) was never a problem. For a detailed analysis of this and other smoothing techniques for Maximum Entropy models, see [18].

4. FEATURE SETS AND EXPERIMENTS

In this section we describe several experiments with the new model, using various feature sets and sampling techniques. We start with the simple reconstruction of n-grams using Gibbs sampling, proceed with longer distance and class based lexical relations using importance sampling, and end with syntactic parse-based features. For subsequent work using semantic features, see [19].

4.1. Validation

To test the feasibility of Gibbs sampling and generally validate our approach, we built a whole-sentence ME model using a small (10K word) training set of Broadcast News [20] utterances[5]. We set $p_0(s)$ to be uniform, and used unigram, bigram and trigram features of the form

$$f_\alpha(s) = \#\text{ of times } n\text{-gram } \alpha \text{ occurs in } s \quad .$$

The features were not introduced all at the same time. Instead, the unigram features were introduced first, and the model was allowed to converge. Next the bigram features were introduced, and the model again allowed to converge. Finally the trigram features were introduced. This resulted in faster convergence than in the simultaneous introduction of all feature types. Training was done by Gibbs sampling throughout.

Below we provide sample sentences generated by Gibbs sampling from various stages of the training procedure. Table 2 lists sample sentences generated by the initial model, before any training took place. Since the initial λ's were all set to zero, this is the uniform model. Tables 3 through 5 list sample sentences generated by the converged model after the introduction of unigram, bigram and trigram features, respectively. It can be seen from the example sentences that the model indeed successfully incorporated the information provided by the respective features.

The model described above incorporates only "conventional" features which are easy to incorporate in a simple conditional language mode. This was done for demonstrative purposes only. The model is unaware of the nature or complexity of the features. Arbitrary features can be accommodated with virtually no change in the model structure or the code.

<s> ENOUGH CARE GREG GETTING IF O. ANSWER NEVER </s>
<s> DEATH YOU'VE BOTH THEM RIGHT BACK WELL BOTH </s>
<s> MONTH THAT'S NEWS ANY YOU'VE WROTE MUCH MEAN </s>
<s> A. HONOR WE'VE ME GREG LOOK TODAY N. </s>

Table 2: Sentences generated by Gibbs sampling from an initial (untrained) model. Since all λ's were initialized to zero, this is the uniform model.

<s> WELL VERY DON'T A ARE NOT LIVE THE </s>
<s> I RIGHT OF NOT SO ANGELES IS DONE </s>
<s> I ARE FOUR THIS KNOW DON'T ABOUT OF </s>
<s> C. GO ARE TO A IT HAD SO </s>
<s> OFF THE MORE JUST POINT WANT MADE WELL </s>

Table 3: Sentences generated by Gibbs sampling from a whole-sentence ME model trained on unigram features only.

<s> DO YOU WANT TO DON'T C. WAS YOU </s>
<s> THE I DO YOU HAVE A A US </s>
<s> BUT A LOS ANGELES ASK C. NEWS ARE </s>
<s> WE WILL YOU HAVE TO BE AGENDA AND </s>
<s> THE WAY IS THE DO YOU THINK ON </s>

Table 4: Adding bigram features.

As we mentioned earlier, Gibbs sampling turned out to be the least efficient of all sampling techniques we considered. As we will show next, much larger corpora and many more features can be feasibly trained with the more efficient techniques.

4.2. Generalized n-grams and Feature Selection

In our next experiment we used a much larger corpus and a richer set of features. Our training data consisted of 2,895,000 words (nearly 187,000 sentences) of Switchboard text (SWB) [16]. First, we constructed a conventional trigram model on this data using a variation of Kneser-Ney smoothing [21], and used it as our initial distribution $p_0(s)$. We then employed features that constrained the frequency of word n-grams (up to n=4), distance-two (i.e. skipping one word) word n-grams (up to n=3) [3], and class n-grams (up to n=5) [22]. We partitioned our vocabulary (of some 15,000 words) into 100, 300, and 1000 classes using the word classing algorithm of [23] on our training data.

[5] Throughout this paper we have been referring to the unit of modeling as a "sentence". But of course, our method can be used to model any word sequence or utterance, whether or not it is consistent with linguistic boundaries. Naturally, linguistically induced features may or may not be applicable to non-sentences.

<s> WHAT DO YOU HAVE TO LIVE LOS ANGELES </s>
<s> A. B. C. N. N. BUSINESS NEWS TOKYO </s>
<s> BE OF SAYS I'M NOT AT THIS IT </s>
<s> BILL DORMAN BEEN WELL I THINK THE MOST </s>
<s> DO YOU HAVE TO BE IN THE WAY </s>

Table 5: Adding trigram features.

feature	training corpus count	trigram corpus count	χ^2
TALKING TO YOU KNOW	0	148	43512.50
TALKING TO _ KNOW	0	148	43512.50
TALKING/CHATTING TO YOU KNOW	0	148	43512.50
NICE/HUMONGOUS TALKING/CHATTING TO YOU KNOW	0	60	7080.50
HOW ABOUT YOU KNOW	0	56	6160.50
HOW ABOUT _ KNOW	0	56	6160.50
<s> HAVE _ KNOW	0	42	3444.50
KIND OF A WHILE/SUDDEN	0	42	3444.50
VAGUELY/BLUNTLY	15389	22604	3382.69

Table 6: n-grams with largest discrepancy (according to χ^2 statistic) between training corpus and trigram-generated corpus of same size; n-grams with "_" token are distance-two n-grams; w_1/w_2 notation represents a class whose two most frequent members are w_1 and w_2

To select specific features we devised the following procedure. First, we generated an artificial corpus by sampling from our initial trigram distribution $p_0(s)$. This "trigram corpus" was of the same size as the training corpus. For each n-gram, we compared its count in the "trigram corpus" to that in the training corpus. If these two counts differed significantly (using a χ^2 test), we added the corresponding feature to our model.[6] We tried thresholds on the χ^2 statistic of 500, 200, 100, 30, and 15, resulting in approximately 900, 3,000, 10,000, 20,000 and 52,000 n-gram features, respectively. n-grams with zero counts were considered to have 0.5 counts in this analysis.

In Table 6, we display the n-grams with the highest χ^2 scores. The majority of these n-grams involve a 4-gram or 5-gram that occurs zero times in the training corpus and occurs many times in the trigram corpus. These are clear examples of longer-distance dependencies that are not modeled well with a trigram model. However, the last feature is a class unigram, and indicates that the trigram model overgenerates words from this class. On further examination, the class turned out to contain a large fraction of the rarest words. This indicates that perhaps the smoothing of the trigram model could be improved.

[6] The idea of imposing a constraint that is most violated by the current model was first proposed by Robert Mercer, who called it "nailing down".

χ^2 threshhold	*baseline*	300	30	15
# features	0	3,500	19,000	52,000
WER	36.53	36.49	36.37	36.29
LM only	40.92	40.95	40.68	40.46
avg. rank	27.29	27.26	26.34	26.42
LM only	35.20	35.28	34.59	33.93

Table 7: Top-1 WER and average rank of best hypothesis using varying feature sets.

For each feature set, we trained the corresponding model after initializing all λ_i to 0. We used importance sampling to calculate expectations. However, instead of generating an entirely new sample for each iteration, we generated a single corpus from our initial trigram model, and re-weighted this corpus for each iteration using importance sampling. (This technique may result in mutually inconsistent constraints for rare features, but convergence can still be assured by reducing the step size F_i with each iteration.) We trained each of our feature sets for 50 iterations of iterative scaling; each complete training run took less than three hours on a 200 MHz Pentium Pro computer.

We measured the impact of these features by rescoring speech recognition N-best lists ($N \leq 200$) which were generated by the Janus system [24] on a Switchboard/CallHome test set of 8,300 words. The trigram $p_0(s)$ served as a baseline. For each model, we computed both the top-1 word error rate and the average rank of the least errorful hypothesis. These figures were computed first by combining the new language scores with the existing acoustic scores, and again by considering the language scores only. Results for the three largest feature sets are summarized in Table 7 (for the smaller feature sets improvement was smaller still). While the specific features we selected here made only a small difference in N-best rescoring, they serve to demonstrate the extreme generality of our model: Any computable property of the sentence which is currently not adequately modeled can (and should) be added into the model.

4.3. Syntactic Features

In the last set of experiments, we used features based on variable-length syntactic categories to improve on an initial trigram model in the Switchboard domain. Our training dataset was the same Switchboard corpus used in Section 4.2.

Due to the often agrammatical nature of Switchboard language (informal, spontaneous telephone conversations), we chose to use a shallow parser that, given an utterance, produces only a flat sequence of syntactic constituents. The syntactic features were then defined in terms of these constituent sequences.

4.3.1. The Shallow Switchboard Parser

The shallow Switchboard parser [25] was designed to parse spontaneous, conversational speech in unrestricted domains. It is very robust and fast for such sentences. First, a series of preprocessing steps are carried out. These include eliminating word repetitions, expanding contractions, and cleaning disfluencies. Next, the parser assigns a part-of-speech tag to each word. For example, the input sentence

Okay I uh you know I think it might be correct

will be processed into

I/NNP think/VBP it/PRPA might/AUX be/VB correct/JJ

As the next step, the parser breaks the preprocessed and tagged sentence into one or more *simplex clauses*, which are clauses containing an inflected verbal form and a subject. This simplifies the input and makes parsing more robust. In our example above, the parser will generate two simplex clauses:

simplex 1: *I/NNP think/VBP*
simplex 2: *it/PRPA might/AUX be/VB correct/JJ*

Finally, with a set of handwritten grammar rules, the parser parses each simplex clause into constituents. The parsing is shallow since it doesn't generate embedded constituents; i.e., the parse tree is flat. In the example, simplex 1 has two constituents:

[_np] ([NP_head] I/NNP)
[_vb] ([VP_head] think/VBP)

and simplex 2 has three constituents:

[_np] ([NP_head] it/PRPA)
[_vb] (might/AUX [VP_head] be/VB)
[_prdadj] (correct/JJ)

The parser sometimes leaves a few function words (e.g. *to, of, in*) unparsed in the output. For the purpose of feature selection, we regarded each of these function words as a constituent. Counted this way, there are a total of 110 constituent types.

4.3.2. Feature Types

As mentioned above, the shallow parser breaks an input sentence into one or more simplex clauses, which are then further broken down into flat sequences of constituents. We defined three types of features based solely on the constituent types; i.e., we ignored the identities of words within the constituents:

1. Constituent Sequence features: for any constituent sequence x and simplex clause s, $f_x(s)$=1 if and only if the constituent sequence of simplex clause s exactly matches x. Otherwise $f_x(s)$=0. For instance, $f_{\text{\_np\_vb}}(\text{I THINK}) = 1$, $f_{\text{\_np\_vb\_prdadj}}(\text{IT MIGHT BE CORRECT}) = 1$, $f_{\text{\_np\_vb}}(\text{IT MIGHT BE CORRECT}) = 0$, and so forth.

2. Constituent Set features: for any set x of constituents, $f_x(s)$=1 if and only if the constituent *set* of sentence s exactly matches x. Otherwise $f_x(s)$=0. This set of features is a relaxation of Constituent Sequence features, since it doesn't require the position and number of constituents to match exactly. As an example, both $f_{\text{\_np,\_vb}}(\text{I LAUGH}) = 1$ and $f_{\text{\_np,\_vb}}(\text{I SEE A BIRD}) = 1$, although the constituent sequence of I LAUGH is *_np_vb* while that of I SEE A BIRD is *_np_vb_np*.

3. Constituent Trigram features: for any ordered constituent triplet $(c1, c2, c3)$, $f_{(c1,c2,c3)}(s)$=1 if and only if sentence s contains that contiguous sequence at least once. Otherwise $f_{(c1,c2,c3)}(s)$=0. This set of features resembles traditional class trigram features, except that they correspond to a variable number of words.

4.3.3. Feature Selection

We followed the procedure described in Section 4.2 to find useful features. We generated an artificial corpus, roughly the same size as the training corpus, by sampling from $p_0(s)$. We ran both corpora through the shallow parser and counted the occurrences of each candidate feature. If the number of times a feature was active in the training corpus differed significantly from that in the artificial corpus, the feature was considered important and was incorporated into the model. Our reasoning was that the difference is due to a deficiency of the initial model $p_0(s)$, and that adding such a feature will fix that deficiency.

We assumed that our features occur independently, and are therefore binomially distributed. More precisely, we had two independent sets of Bernoulli trials. One is the set of n simplex clauses of the training corpus. The other is the set of m simplex clauses of the artificial corpus. Let x be the number of times a feature occurs in the training corpus and y that in the artificial corpus. Let P_x and P_y be the

true occurrence probabilities associated with each corpus. We tested the hypotheses $H_0 : P_x = P_y$. Approximating the Generalized Likelihood Ratio test, we rejected H_0 at confidence level α if

$$\left| \frac{\frac{x}{n} - \frac{y}{m}}{\sqrt{\left(\frac{x+y}{m+n}\right) \cdot \left(1 - \frac{x+y}{m+n}\right) \cdot \left(\frac{n+m}{nm}\right)}} \right| \geq z_{\alpha/2}$$

(see for example [26, page 335]). We incorporated into our model those features whose H_0 was rejected.

4.3.4. Results

Constituent Sequence features: There were 186,903 candidate features of this type that occurred at least once in the two corpora. Of those, 1,935 show a significant difference between the two corpora at a 95% confidence level (two-tailed). The feature $f_{\_np\_vb\_np\_pp}$ had the most significant standard score 21.9 in the test, with x=2968 occurrences in the SWB corpus and y=1548 in the artificial corpus. More interesting is the feature $f_{\_conj\_np\_aux\_prdadj}$, with z-score 4.3, and x=0, y=19. One may suspect that this is where the initial trigram model "makes up" some unlikely phrases. Looking at the 19 simplex clauses confirms this:

SO I HAVE NEVER REALLY INTERESTING
AND THEY MIGHT PRACTICAL
THAT WE HAVE GOOD
THAT YOU COULD LAST
BUT I WOULD SURE
AND YOU CAN CONVENIENT
. . .

Similarly, the feature $f_{\_wh\_np\_vb\_np\_vb\_in}$ has standard score -4.0, x=16 and y=0. This feature stands for a perfectly plausible simplex clause form that has never been generated in the artificial corpus, probably because it contains a long-distance dependence. Indeed, the corresponding simplex clauses in SWB are:

WHAT AREA DO YOU WORK IN
WHAT AREA DO YOU LIVE IN
WHAT HOME DO YOU LIVE IN
WHAT EXERCISE DO YOU GET INVOLVED IN
. . .

Constituent Set features: These features are more general than Constituent Sequence features and thus there are fewer of them. A total of 61,741 candidate Constituent Set features occurred in either corpus, while 1310 showed a significant difference. The one with the most significant z-score, 27.8, is $f_{\_conj,\_np,\_pp,\_vb}$, with x=10420 and y=6971. Like Constituent Sequence features, there were some Constituent Set features that occurred only in the artificial corpus. For example, $f_{\_adv,\_conj,a}$ had a z-score of 4.0 with x=0 and y=16:

OR A TOTALLY
AND A PROPERLY
IF A WHATSOEVER
. . .

There were also features that only occurred in the SWB corpus, such as $f_{\_np,\_pp,\_vb,\_wh,from}$ with z-score 3.8, x=14 and y=0.

Constituent Trigram features: 36,448 candidate features of this type appeared in the corpora, of which 3,535 were significant. The feature $f_{\_np,\_adv,some}$ with z-score 4.9, x=0 and y=25 is another good example of the deficiencies of the initial trigram model:

BUT HE NEVER SOME
WE THE GYM EVEN SOME
IT REALLY SOME REALLY BAD
MYSELF SOMETIMES SOME ON CHANNEL 8
DOLLARS
. . .

4.3.5. Perplexity and Word Error Rate

We incorporated the 1953 Constituent Sequence features, 1310 Constituent Set features, and 3535 Constituent Trigram features into a whole-sentence maximum entropy language model, and trained its parameters with the GIS algorithm. The baseline perplexity of a 90,600-word SWB test set calculated under the initial model p_0 was 81.37. The perplexity under the new maximum entropy model was estimated as 80.49$\pm$0.02, a relative improvement of only 1%.

Next, we tested speech recognition word error rate by N-best list rescoring. A 200-best list with 8,300 words was used. The WER was 36.53% with the initial model and 36.38% with all of the syntactic features added, a mere 0.4% relative improvement.

5. ANALYSIS

In trying to understand the disappointing results of the last section, we analyzed the likely effect of features on the final model. The upper bound on improvement from a single binary feature f_i is the Kullback Liebler distance between the true distribution of f_i (as estimated by the empirical distribution $\tilde{p}(f_i)$) and $p(f_i)$ (the distribution of f_i according to the current model) [14, p. 4]. The effect of multiple features is not necessarily additive (in fact, it could be supra- or sub-additive). Nonetheless, the sum of the individual effects may still give some indication of the likely combined

effect. For the syntactic features we used, we computed:

$$\sum_i D(\tilde{p}(f_i)||p_0(f_i)) = 0.062$$

which translates into an expected perplexity reduction of 0.43% ($2^{\frac{0.062}{10}}$, where 10 is the average number of words in a sentence). The potential impact of these features is apparently very limited. We therefore need to seek features f for which:

$$D(\tilde{p}(f)||p_0(f)) = \tilde{p}(f) \cdot \log \frac{\tilde{p}(f)}{p_0(f)} + (1-\tilde{p}(f)) \cdot \log \frac{1-\tilde{p}(f)}{1-p_0(f)}$$

is significantly larger. The second term on the right-hand side is usually negligible. The two factors affecting this number are thus the prevalence of the feature ($\tilde{p}(f)$) and the log discrepancy between the truth and the model ($\log \frac{\tilde{p}(f)}{p_0(f)}$). In the features we used, the latter was quite large, but the former was very small. Thus, we need to concentrate on more common features.

An ideal feature should occur frequently enough, yet exhibit a significant discrepancy. "Does the sentence make sense to a human reader?" is such a feature (where $\tilde{p}(f) \approx 1$ and $p_0(f) \approx 0$). It is, of course, AI-hard to compute. However, even a rough approximation of it may be quite useful. Based on this analysis, we have subsequently focused our attention on deriving a smaller number of frequent (and likely more complex) features, based on the notion of sentence coherence ([27]).

Frequent features are also computationally preferable. Because the training bottleneck in whole-sentence ME models is in estimating feature expectations via sampling, the computational cost is determined mostly by how *rare* the features are and how *accurately* we want to model them. The more frequent the features, the less the computation. Note that computational cost of training depends much less on the vocabulary, the amount of training data, or the number of features.

6. SUMMARY AND DISCUSSION

We presented an approach to incorporating arbitrary linguistic information into a statistical model of natural language. We described efficient algorithms for constructing whole-sentence ME models, offering solutions to the questions of sampling, step size and smoothing. We demonstrated our approach in two domains, using lexical and syntactic features. We also introduced a procedure for feature selection which seeks and exploits discrepancies between an existing model and the training corpus.

Whole-sentence ME models are more efficient than conditional ME models, and can naturally express sentence-level phenomena. It is our hope that these improvements will break the ME "usability barrier" which heretoforth hindered exploration and integration of multiple knowledge sources. This will hopefully open the floodgates to experimentation, by many researchers, with varied knowledge sources which they believe to carry significant information. Such sources may include:

- Distribution of verbs and tenses in the sentence
- Various aspects of grammaticality (person agreement, number agreement, parsability, other parser-supplied information)
- Semantic coherence
- Dialog level information
- Prosodic and other time related information (speaking rate, pauses,...)

Since all knowledge sources are incorporated in a uniform way, a language modeler can focus on *which* properties of language to model as opposed to *how* to model them. Attention can thus be shifted to feature induction. Indeed, we have started working on an interactive feature induction methodology, recasting it as a logistic regression problem [27, 19]. Taken together, we hope that these efforts will help open the door to "putting language back into language modeling" [28].

Acknowledgements

We are grateful to Sanjeev Khudanpur, Fred Jelinek and Prakash Narayan for helpful discussions during the early stages of this work; to Larry Wasserman for much advice on sampling techniques; to Klaus Zechner for use of his parser; and to the reviewers for many thoughtful suggestions and comments.

7. REFERENCES

[1] Ciprian Chelba and Fred Jelinek. (to be published in csl oct 00). *Computer Speech and Language*, 15(??):??–??, 2000.

[2] Sanjeev Khudanpur and Jun Wu. Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling. *Computer Speech and Language*, to appear.

[3] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228, 1996. longer version published as "Adaptive Statistical Language Modeling: A Maximum Entropy Approach," Ph.D. thesis, Computer Science Department, Carnegie Mellon University, TR CMU-CS-94-138, April 1994.

[4] Ronald Rosenfeld. A whole sentence maximum entropy language model. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

[5] Stanley F. Chen and Ronald Rosenfeld. Efficient sampling and feature selection in whole sentence maximum entropy language models. In *ICASSP-99*, Phoenix, Arizona, 1999.

[6] Xiaojin Zhu, Stanley F. Chen, and Ronald Rosenfeld. Linguistic features for whole sentence maximum entropy language models. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Budapest, Hungary, 1999.

[7] E.T. Jaynes. Information theory and statistical mechanics. *Physics Reviews*, 106:620–630, 1957.

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

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

[10] 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.

[11] Raymond Lau, Ronald Rosenfeld, and Salim Roukos. Trigger-based language models: A maximum entropy approach. In *Proceedings of ICASSP-93*, pages II–45 – II–48, April 1993.

[12] Chris Paciorek and Roni Rosenfeld. Minimum classification error training in exponential language models. In *Proceedings of the NIST/DARPA Speech Transcription Workshop*, May 2000.

[13] S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions, and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:721–741, 1984.

[14] S. Della Pietra, V. Della Pietra, and J. Lafferty. Inducing features of random fields. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4):380–393, April 1997.

[15] N. Metropolis, A.W. Rosenbluth, M.N. Rosenbluth, A.H. Teller, and E. Teller. Equations of state calculations by fast computing machines. *Journal of Chemical Physics*, 21:1087–1092, 1953.

[16] J.J. Godfrey, E.C. Holliman, and J. McDaniel. SWITCHBOARD: Telephone speech corpus for research and development. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 517–520, March 1992.

[17] Stephen Della Pietra and Vincent Della Pietra. Statistical modeling by maximum entropy. unpublished report, 1993.

[18] Stanley F. Chen and Ronald Rosenfeld. A survey of smoothing techniques for maximum entropy models. *IEEE Transactions on Speech and Audio Processing*, 8(1):37–50, 2000.

[19] 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.

[20] David Graff. The 1996 broadcast news speech and language model corpus. In *Proceedings of the DARPA Workshop on Spoken Language technology*, pages 11–14, 1997.

[21] 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.

[22] Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jennifer C. Lai, and Robert L. Mercer. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479, December 1992.

[23] Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependences in stochastic language modeling. *Computer Speech and Language*, 8:1–38, 1994.

[24] M. Finke, J. Fritsch, P. Geutner, K. Ries, T. Zeppenfeld, and A. Waibel. The JanusRTk Switchboard/Callhome 1997 evaluation system. In *Proceedings of LVCSR Hub 5-E Workshop*, Baltimore, May 1997.

[25] Klaus Zechner. Building chunk level representations for spontaneous speech. Master's thesis, Carnegie Mellon University, Department of Philosophy, 1997.

[26] R. Larsen and M. Marx. *An introduction to Mathematical Statistics and Its Applications*. Prentice-Hall, NJ, 1981.

[27] Ronald Rosenfeld, Larry Wasserman, Can Cai, and Xiaojin Zhu. Interactive feature induction and logistic regression for whole sentence exponential language models. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, Keystone, CO, December 1999.

[28] Fred Jelinek. The 1995 language modeling summer workshop at Johns Hopkins University. Closing remarks.