INVESTIGATING LINGUISTIC KNOWLEDGE IN A MAXIMUM ENTROPY TOKEN-BASED LANGUAGE MODEL

Jia Cui, Yi Su, Keith Hall and *Frederick Jelinek*

Center for Language and Speech Processing
The Johns Hopkins University, Baltimore, MD, USA
{cuijia,suy,keith_hall,jelinek}@jhu.edu

ABSTRACT

We present a novel language model capable of incorporating various types of linguistic information as encoded in the form of a *token*, a (word, label)-tuple. Using tokens as hidden states, our model is effectively a hidden Markov model (HMM) producing sequences of words with trivial output distributions. The transition probabilities, however, are computed using a maximum entropy model to take advantage of potentially overlapping features. We investigated different types of labels with a wide range of linguistic implications. These models outperform Kneser-Ney smoothed n-gram models both in terms of perplexity on standard datasets and in terms of word error rate for a large vocabulary speech recognition system.

1. INTRODUCTION

Statistical language models (LM) represent a probability distribution over sequences of words, usually making sequential decisions from left to right, each prediction dependent on a limited context. The main challenge comes from data sparseness: many sequences in the test data are unseen in the training data. Data clustering has shown to be efficient in addressing this problem. In widely used n-gram language models, histories are clustered if they end in the same $(n-1)$ words.

Previously, there has been some success at incorporating the use of word equivalence classes into language modeling [1, 2]. In these models, words are assigned to classes independent of the context. But in natural language, a word expresses different properties in different contexts. Additionally, correctly understanding the semantic and syntactic function of each word influences the likelihood of observing a particular whole sentence. In this paper, we propose a token-based LM, where tokens are tuples of words and associated labels. This model accommodates not only word equivalence classes but also arbitrary contextually-restricted word labels. The new challenge is that the labels are unknown at test time. Our model simply computes the marginal distribution of the word sequence, effectively summing over all label sequences possible for the test data.

We introduce the Maximum Entropy Token-based Language Model (METLM) in Section 2, and then discuss parameter estimation and inference algorithms in Section 4. Empirical results, evaluated both in terms of perplexity and in word error rate (WER) for a state-of-the-art speech recognizer, are presented in Section 5, followed by conclusions.

2. MAXIMUM ENTROPY TOKEN-BASED LANGUAGE MODEL

We encode linguistic knowledge in the form of word *labels* which can be context dependent. One word can be attached with multiple labels, each reflecting different properties of the word or its context. For example, the word 'football' in the sentence 'he loves to play football' can be labeled both semantically as a 'SPORT' and syntactically as a 'NOUN'.

In this work, we define a *token* as a $(\text{word}, \text{label})$ pair. For simplicity, in this article all derivations assume one label per word occurrence; however, multiple labels can be applied using the same principle. We call a word *ambiguous* if it is part of multiple possible tokens associated with it, that is, it can have different labels in different contexts. If all words are unambiguous, the probability of a word sequence w_1^m is simply

$$p(w_1^m) = p(w_1^m, l_1^m) = \prod_{i=1}^{m} p(w_i, l_i | w_1^{i-1}, l_1^{i-1})$$

In the general case where some words are ambiguous, the probability of a word sequence is the sum over probabilities of all its possible token sequences (i.e., we marginalize over token sequences):

$$p(w_1^m) = \sum_{l_1^m} \prod_{i=1}^{m} p(w_i, l_i | l_1^{i-1}, w_1^{i-1}) = \sum_{l_1^m} \prod_{i=1}^{m} p(s_i | s_1^{i-1}), \quad (1)$$

where we use $s_i = (w_i, l_i)$ to denote a token. Figure 1 shows a token trellis with bigram dependencies. In this example, three words in the sentence have multiple possible POS tags, therefore, we calculate probabilities of all eight possible token paths of the sentence at test time.

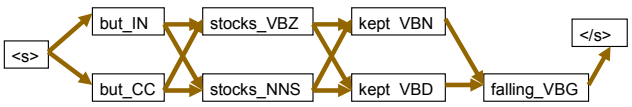

Fig. 1. An example of token trellis for a sentence

As in word-based n-gram LMs, we assume a simple Markov process. We use a maximum entropy model for state transition probabilities:

$$p(s_i|s_{i-n+1}^{i-1}) = \frac{\exp(\sum_k \lambda_k f_k(s_{i-n+1}^{i}))}{Z(s_{i-n+1}^{i-1})} \quad (2)$$

where λ_k is a real-valued parameter, Z is a normalization variable which depends only on the n-gram token history, and f_k is a binary feature function. For instance, $f(w_{i-1} = \text{kept}, l_i = \text{VBG})$ equals 1 if and only if the word in position $(i-1)$ is 'kept' and the future word is labeled as 'VBG'.

In maximum entropy (ME) modeling, each feature is associated with a constraint. The overall constraint set determines the model and reflects our understanding of the observed data. For example, features in the form of $f(l_{i-1}, w_i)$ imply that the distribution of w_i depends on the label in position $(i-1)$. Table 1 shows some feature types and their descriptions. For each position in an n-gram feature, we take either the word or the label at that position instead of both. This avoids further data sparseness because label-based features have empirical frequencies no lower than those of the corresponding n-gram word features. Moreover, the label-based features address data sparseness by classifying words into different syntactic groups. In our experiments, all thresholds for features by default are zero, that is, as long as a label/word n-gram appears in the training data and its type is included, the n-gram is used to form a feature.

Type	Description
W	unigram word feature. $f(w_i)$
WW	bigram word feature. $f(w_{i-1}, w_i)$
WWW	trigram feature. $f(w_{i-2}, w_{i-1}, w_i)$
TW	bigram feature. $f(l_{i-1}, w_i)$
WTW	trigram feature. $f(w_{i-2}, l_{i-1}, w_i)$
TWW	trigram feature. $f(l_{i-2}, w_{i-1}, w_i)$
TTW	trigram feature. $f(l_{i-2}, l_{i-1}, w_i)$
T	unigram label feature. $f(l_i)$
W:T	composite unigram feature. $f(w_i, l_i)$
WT	bigram feature. $f(w_{i-1}, l_i)$
TT	bigram feature. $f(l_{i-1}, l_i)$
WWT	trigram feature. $f(w_{i-2}, w_{i-1}, l_i)$
...	...

Table 1. Feature Types

3. RELATED WORK

The main difference between our models (METLMs) and traditional ME LMs [3, 4] is that our model predicts tokens instead of words. This change enhances language modeling in several aspects. First, the new model enables us to integrate ambiguous word labels into language modeling. We can infer the hidden word labels during test while the traditional models can only model explicit word labels used in the conditioning context. Second, the new model can integrate future label information directly. Finally, the new framework can be applied for unsupervised training.

We have built an LM based on tokens and derived a parameter estimation algorithm based on the statistics of token elements. The concept of a token is similar to the superset in SuperARV LM [5] and the factor vector in the factored LM (FLM) [6]. The underlying models are quite different. While they use backoff smoothing techniques to model a conditional distribution, we apply the maximum entropy principle to integrate features naturally by a log-linear model.

4. PARAMETER ESTIMATION AND INFERENCE

When all words are unambiguous, i.e., each word is associated with one label, the training and test process is straightforward: we simply label both the training and test datasets. In training, we build a model and estimate parameters by maximizing the likelihood of the labeled training data. At test time, we simply predict the token based on the unambiguous token histories. The advantage of including labels is that we can have features like $f(l_{i-1}, w_i)$ which can help alleviate data sparseness.

The model becomes more complicated when a word can take on multiple labels. First, we describe the procedure for the case where we have labeled training data. In training, we build a model and optimize it to maximize the joint likelihood of the *labeled* training data, that is, the observed token sequence: $L_\Lambda = \log p(w_1^m, l_1^m; \Lambda) + \log p(\Lambda|\Delta)$ where Λ denotes the feature set and Δ denotes the Gaussian prior [7]. The feature parameters are estimated using the Improved Iterative Scaling algorithm [3] equipped with the speed-up method proposed in [8].

It is also possible to train the model with unlabeled training data. With unlabeled training data, the goal is to maximize the marginal likelihood of training data using the latent labels: $L_\Lambda = \log \sum_{l_1^m} p(w_1^m, l_1^m; \Lambda) + \log p(\Lambda|\Delta)$. The model we have is simply an HMM with fixed output distributions and can be trained via EM [9]. In the E-step, expected counts for each transition are added to the expectations of features activated by this transition. Expected token counts are accumulated during the forward algorithm. These expectations develop updated constraints. The M-step calculates new feature parameters for the next iteration with an embedded ME training procedure that uses the updated constraints. This EM

algorithm is guaranteed to converge. In the empirical section of this work, we present results only for labeled training data because the unsupervised training is too computationally expensive.

The test data probability can be obtained by a single pass using the forward algorithm. It sums over probabilities of all possible token paths of the test data. In this formulation, the prediction of each word w_i is computed as follows:

$$p(w_i|w_1^{i-1}) = \frac{p(w_1^i)}{p(w_1^{i-1})} = \frac{\sum_{l_1^i} p(w_1^i, l_1^i)}{\sum_{l_1^{i-1}} p(w_1^{i-1}, l_1^{i-1})} \quad (3)$$

5. EXPERIMENTAL RESULTS

5.1. Experimental Setup

The dataset we have used to evaluate the perplexity performance is from the UPenn Treebank-3 [10]: the parsed Wall Street Journal (WSJ) collection. All words are lowercased and all punctuation is removed. Numbers are substituted by a special word 'N'. An open vocabulary consisting of 10K words with an extra 'UNK' word is used. The WSJ corpus contains 24 sections. The first 20 sections are taken as training data, containing 1M words; the following two sections are used as held-out data for setting model hyper-parameters, and the last two sections are test data.

Our baseline model used the modified Kneser-Ney smoothing [11] without any word classes and was built with the SRI LM toolkit [12]. We first trained a METLM model with word features only, i.e., by ignoring any label information and tuned the three feature priors on the held-out data (one prior for each n-gram). The result was comparable to that of the baseline model as [7] observed. The baseline perplexity for this model on the test-set was **144**.

We also built the second baseline by interpolating several class-based LMs with the dominant POS tags (effectively making the labels unambiguous). The baseline model was the interpolation of four Kneser-Ney smoothed LMs. We extracted counts for word/label n-grams: WWW, WTW, TWW and TTW. For example, counts associated with the WTW feature type contained counts of (w_{i-2}, l_{i-1}, w_i), (l_{i-1}, w_i) and (w_i). For each feature type, we built a smoothed trigram LMs using the modified Kneser-Ney smoothing (we trained using the SRI LM toolkit). Correspondingly, for each test event (w_1, w_2, w_3), we first labeled all words with their dominant POS tags and then generated four probabilities $p(w_3|w_1, w_2)$, $p(w_3|l_1, w_2)$, $p(w_3|w_1, l_2)$ and $p(w_3|l_1, l_2)$ with corresponding LMs. The four sets of scores were interpolated to get the perplexity **138**.

Using the priors optimized for the word-based models, we then introduced label-based features. Since we fixed the priors and no longer needed to tune hyper-parameters, we included the held-out data in our training set and trained the model again.

5.2. POS Tags and Data-Driven Word Classes

In our first set of experiments, we explored the modeling effect, evaluated by perplexity, of using various types of word classes. We used human-annotated POS tags from the Treebank (truePOS) as well as the dominant POS tags (domiPOS). The test procedure for truePOS and domiPOS were quite different. In the former, we considered all possible POS sequences for the test sentences and summed over them; in the later, we assumed each word could only be assigned the dominant POS tag and therefore only one POS sequence was available for each test sentence. For comparison, we also trained models on the training data labeled with position-dependent word classes [2] (PD-CLS), where different classes are generated for different positions using an exchange algorithm, as well as position-independent classes based on the co-occurrence of word pairs [1] (PI-CLS). For position-dependent word classes, we generated 64 classes at each position [1]. That means for each word w_i, there were three labels l_i^0, l_i^{-1} and l_i^{-2}. These labels were used to compose different types of features according to their positions in the feature. For example, in extracting TWT features, we used the trigram $(l_{i-2}^{-2}, w_{i-1}, l_i^0)$. For PI-CLS, we simply generated 64 classes using the SRI LM toolkit.

Table 2 reports the perplexity for models trained with different word labels and different feature sets. The first column shows the types of label-based features (denotations of feature types are explained in Table 1) included in modeling. 'TW' means TW features are included in the model. 'WT+' means WT, T and W:T features are used in the model. 'AA' means T, W:T, TW, WT, TT features are included. 'All' means T, W:T, WT, TW, TT, WTW, WWT, TWT, TTW, WTT features are included. 'hisT' means TW, WTW, TWW and TTW features are included. Note that basic word features W,WW,WWW are included in every model.

Feature	PI-CLS	PD-CLS	domiPOS	truePOS
TW	138	138	137	146
WTW	141	142	139	143
TWW	142	144	143	144
WT+	138	138	137	136
TW,WT+	135	138	134	132
WTW,WT+	136	135	133	132
AA	134	137	133	131
AA, WTW	133	132	130	128
All	129	131	126	**122**
hisT	138	138	131	N/A

Table 2. Perplexity on UPenn WSJ corpus

Generally, labels were helpful. As more label-based features were added to the model, the performance improved.

[1] The number of classes was selected based on the heldout data.

Most models performed better than the modified Kneser-Ney baseline which used no label information. Of all different labels, the true POS tag technique improved prediction greatest with perplexity dropping to as low as 122.

In the first group of experiments in Table 2, we tested each single type of label-based features. Bigram features, TW and WT+, improved performance more because they helped more predictions than the trigram features WTW and TWW. The second group of experiments showed the performances of different combinations. Generally, more features lead to better performance.

Note that although we used true POS in the training data, we did not use any labeling information from the test data. Our model computed the labeling sequence distribution over the plain test sentences. To make it clearer, note that results of using only TW, WTW or TWW features for the true POS tag set were not improved over the Kneser-Ney baseline. This is because the POS tags are determined mostly by the word being predicted rather than the neighboring labels and words. Excluding future labels in features leads to low-quality label distributions during the test, and therefore contributes little to the model.

As we have mentioned in Section 4, our model is different from the traditional ME model in that our model predicts tokens instead of words. This difference enables us to explore future labels in language modeling (meaning the label of the word being predicted). We have shown the importance of future labels in the ambiguous case (truePOS). Here, we emphasize the importance of including future labels in language modeling by building models excluding these future labels in the feature set. The results are displayed in Table 2 row 'hisT'. These results are comparable to our second baseline obtained by interpolating four Kneser-Ney smoothed LMs using the same types of features. Even using unambiguous labels (domiPOS and PD-CLS classes), including future label-related features leads to a decent improvement over excluding those features.

5.3. Language Modeling with Different Word Labels

In this subsection, we compare the effects of word labels generated from processes intended to capture different linguistic categories. The word classes used above, the position-independent and the position-dependent word classes, are determined based on the neighboring two or four words. Here we introduce three additional data-driven word classes that are generated from sentential contexts and document information.

First, we experiment with the *dependency-based* word classes of Dekang Lin [13]. A dependency relationship [14] is an asymmetric binary relationship between a word and its semantic/syntactic dependents; these are called the *head* and *modifier*, respectively. Figure 2 shows an example of dependency tree with links from the head to the modifiers (c.f. [15]). There are three sets of dependency-based classes for words belonging to nouns, verbs and adjectives respectively. In the POS-labeled training data, we use the corresponding word classes as the word labels and form features based on these labels.

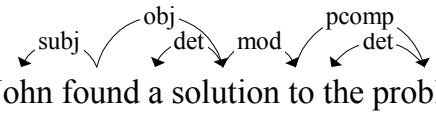

Fig. 2. An example of dependency relationship

We also experiment with Lin's *proximity-based* word classes [13]. These are based solely on the linear proximity relationship between words. Labels based on these classes are unambiguous because each word belongs to only one class.

Finally, we consider the *topic-based* word classes as described in [16]. This is a vector-based topic model where each word is represented by a vector in a lower dimensional semantic feature space. The distance between any two words is the cosine distance of the corresponding two word vectors. Two words are likely to be clustered together if they tend to be observed in similar documents, regardless of their syntactic roles.

We obtained the dependency-based and proximity-based class data from Lin [17] and the topic-based data from Deng and Khudanpur [16]. Using the similarity scores assigned under each model, we applied a bottom-up, agglomerative word clustering algorithm in order to generate equivalence classes. The algorithm initially treats each word as its own class and then merges the two classes which are closest. These classes are merged and then the processes repeats. We continue the process until we are left with 100 classes. In order to measure the distance between two classes, we take the average distance between a bipartite mapping of words contained in the two classes.

Motivated by the good performance of POS tags, we added an experiment using the POS tag of the head-word as word labels. For example, the sentence in Figure 2, the word 'found' is the head of the word 'John' and the word 'found' has the POS tag 'VBD', therefore the label for 'John' is 'VBD'[2].

We labeled the training data with head-word POS tags (headPOS), proximity-based word classes (wordProx), dependency-based word classes (wordDepen) and topic-based word classes (wordTopic) respectively and built four METLMs with the T, W:T, TW, WT, TT, WTW features (Table 1)[3]. The results for these experiments can be found in Table 3.

The results, in terms of perplexity, for the models with varying word classes are all very similar. Improvements over the Kneser-Ney smoothed generative model is relatively small.

[2]We chose to use the POS tag of the head rather than the head word itself in order to keep the decoding trellis manageable.

[3]In each of these experiments, we used only with one particular feature set to offer a comparison between models with different word classes.

Classes	Perplexity
Kneser-Ney	144
PI-CLS	133
PD-CLS	132
domiPOS	130
truePOS	128
headPOS	139
wordProx	136
wordDepen	137
wordTopic	139

Table 3. Perplexities with different word labels

The dependency-based word class model's relative improvement is worth pointing out as only about one third of the words had valid classes. [4]

5.4. Feature Selection

In this subsection, we present two intuitive methods for threshold-based feature selection in METLM. We start with a detailed inspection of the perplexity improvements by considering specific partitions of test data. We partitioned all predictions in the test data by the occurrence counts of histories as observed in the training data. Then we calculated the perplexity $\exp(\frac{1}{K}\sum_1^K \log p(w_k|h_k))$ for each partition where K is the total number of predictions in that partition.

The result of the position-dependent word class model (Table 2 row AA, WTW; column PD-CLS), are partitioned and presented in Table 4 Column PD3. To its left is the partitioned trigram Kneser-Ney smoothing result (KN3). To its right we present results for the 5-gram Kneser-Ney smoothing LM (KN5). $c(h) = c(w_{i-2}, w_{i-1})$ is the history count in the training data and $c(\overline{h}) = c(w_{i-1})$ is the backoff history count. Predictions are assigned to the first row where the condition is met. The first column (PER) is the percentage of prediction counts in the test data.

Category	PER	KN3	PD3	KN5	PD3+
$C(h) > 50$	25	114	109	100	99
$C(h) > 0$	41	129	116	125	115
$C(\overline{h}) > 0$	27	177	159	174	160
Others	6	310	305	280	295
Total	100	144	132	137	129

Table 4. Perplexities in different partitions of test data

Comparing column PD3 and KN5 with the baseline KN3 in Table 4, note that PD3 achieves a greater improvement in predictions with infrequent histories. For predictions with frequent histories, long-history features (KN5) are more helpful. We then built the PD3+ model (last column in Table 4) with the feature set from PD3. A subset of the 4 and 5-gram word features. w_{i-3}^i and w_{i-4}^i from the training data were selected as features if and only if the trigram history count (w_{i-2}, w_{i-1}) was over 50. This additional feature set comprised only 20% of all 4 and 5-grams in the training data. Most of these selected features appeared only once. But the selected 4 and 5-gram observations contributed most of the improvement achievable by the complete set of 4 and 5-grams in KN5.

The above experiment suggests a new method for setting thresholds in feature selection for language modeling. The threshold is set not based on the absolute count of the feature itself, but on the frequency of the suffix of the history component.

Similarly, we have considered setting thresholds for infrequent, redundant features. The basic idea is: if an n-gram feature appears only once, there is no need to add related higher-order n-gram features. To be more specific, if (w_{i-1}, w_i) appears only once, we remove $w_{i-k}^i, k > 1$ from the feature set. This principle led to a 20% reduction of trigram features on 1M words of the WSJ Treebank data without affecting the performance. In our word error rate experiments, we applied this method to reduce 20% of the 4-gram features from 20M words of training data.

This second method sets thresholds based on the frequency of the suffix of the feature. This method divides all singleton trigrams into two sets. One set is regarded as redundant and are removed; the other set remains because it contains useful information which is not covered by other features. For example, assume 'keeps falling' and 'keeps rising' occur 5 times each in the training data. 'price keeps falling' occurs once and 'price keeps rising' never occurs. Given the history 'price keeps', the model will prefer 'falling'. This preference will not hold if 'price keeps falling' is filtered out.

5.5. Evaluation by Speech Recognition Performance

In order to determine if the above improvements carry over to actual speech-recognition performance, we tested our LM on a large-vocabulary speech recognition task. We use the IBM conversational telephony system for rich transcription (RT-04 CTS system) [18]. The experiment is conducted on the Fisher data collection (DEV04 English), which contains 36 telephone conversations recorded while two speakers were talking about a randomly chosen topic. It has utterances from 72 speakers and contains 9,044 utterances and 37,834 words. A small LM (trained on 4M words) was used to generate word lattices for this test set.

The IBM RT-04 system used a vocabulary with 30,500 words. Word-lattices were built and then re-scored with a larger language model based on 150M words of data. Four-gram generative language models with the modified Kneser-Ney smoothing were used in the IBM system. The baseline

[4] This is primarily due to the fact that we only label content words.

error rate for the first-pass system (using the small LM) was 14.1%. That score went down to 13.4% after re-scoring with the LM trained on 150M words.

We utilized the dominant POS tags which were generated from 3M words of Switchboard Treebank data to label the training data. We built an METLM including basic word features W, WW, WWW, WWWW and T, W:T, TW, WT, TT, WTW, WWT, WWTW, WTWW features sets. Our model reduces the WER to 13.7% and 13.2% when interpolating it with the original LM. Both improvements were significant with $p_{value} < 0.001$. In this experiment, 4-gram features were filtered according to our second principle of feature selection (Section 5.4).

Model	w/o interpolation	w/ interpolation
KN-4gm	14.1	13.5
METLM-4gm	13.7	13.2

Table 5. Word Error Rates on Fisher Data

6. CONCLUSIONS

We have developed a maximum entropy token-based language model (METLM) which encapsulates words and their latent linguistic labels into tokens and exploits parallel dependencies between components of different tokens. The model integrates all possible local dependencies to help predictions in a straightforward way. We have shown the effectiveness of this model by using only POS tags to achieve substantial relative perplexity reduction (15%) on the UPenn WSJ Treebank data and a significant WER reduction (0.4%) on the Fisher data (DEV04 English) over the standard generative backoff model using modified Knesner-Ney smoothing.

The METLM offers a platform to integrate arbitrary linguistic knowledge which can be represented as word labels. We have carried out experiments with labels generated from local contexts, dependency relationships and document-word co-occurrences. All of these provide useful knowledge in predictions and have proven to outperform the baseline models. Particularly, models based on word labels which are based on local contexts achieve the best performance.

We also presented two new methods of feature filtering by utilizing their hierarchical structure instead of setting thresholds on absolute counts of features themselves in the training data, we filtered out n-gram features based on their lower-order n-gram counts and found them effective in significantly reducing active feature set size while maintaining predictive capabilities.

7. REFERENCES

[1] P. F. Brown, P. V. deSouza, R. L. Mercer, V. J. Della Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational Linguistics*, vol. 18, no. 4, pp. 467–479, 1992.

[2] A. Emami and F. Jelinek, "Random clusterings for language modeling," in *Proc. of ICASSP*, vol. 1, pp. 581–584.

[3] A. L. Berger, S. D. Pietra, and V. J. Della Pietra, "A maximum entropy approach to natural language processing," *Computational Linguistics*, vol. 22, no. 1, pp. 39–71, 1996.

[4] S. Khudanpur and J. Wu, "Maximum entropy techniques for exploiting syntactic,semantic and collocational dependencies in language modeling," *Computer Speech and Language*, vol. 14, no. 4, 2000.

[5] W. Wang and M. P. Harper, "The superarv language model: investigating the effectiveness of tightly integrating multiple knowledge sources," in *Proc. of EMNLP*, 2002, pp. 238–247.

[6] J. Bilmes and K. Kirchhoff, "Factored language models and generalized parallel backoff," in *Proc. of HLT/NACCL*, 2003, pp. 4–6.

[7] S. Chen and R. Rosenfeld, "A gaussian prior for smoothing maximum entropy models," Tech. Rep. CMUCS -99-108, Carnegie Mellon University, 1999.

[8] J. Wu and S. Khudanpur, "Combining nonlocal, syntactic and n-gram dependencies in language modeling," in *Proc. of Eurospeech*, 1999, pp. 2179–2182.

[9] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of markov chains," *The Annals of Mathematical Statistics*, vol. 41, pp. 164–171, 1970.

[10] M. P. Marcus, B. Santorini, and M. A. Marcinkiewicz, "Building a large annotated corpus of english: The penn treebank," *Computational Linguistics*, vol. 19, pp. 313–330, 1993.

[11] S. F. Chen and J. T. Goodman, "An empirical study of smoothing techniques for language modeling," in *Technical Report TR-10-98, Computer Science Group*. 1998, Harvard University.

[12] A. Stolcke, "SRILM – an extensible language modeling toolkit," in *Proc. Intl. Conf. on Spoken Language Processing*, 2002.

[13] D. Lin, "Automatic retrieval and clustering of similar words," in *COLING-ACL*, 1998, pp. 768–774.

[14] D. G. Hays, "Dependency theory: A formalism and some observations," *Language*, vol. 40, pp. 511–525, 1964.

[15] D. Lin and P. Pantel, "Induction of semantic classes from natural language text," in *Proc. of SIGKDD*, 2001, pp. 317–322.

[16] Y. Deng and S. Khudanpur, "Latent semantic information in maximum entropy language models for conversational speech recognition," in *HLT-NAACL*, May 2003, pp. 56–63.

[17] D. Lin, "Proximity-based thesaurus and dependency-based thesaurus," in *http://armena.cs.ualberta.ca/lindek/downloads*, 2000.

[18] H. Soltau, B. Kingsbury, L. Mangu, D. Povey, G. Saon, and G. Zweig, "The IBM 2004 Conversational Telephony System for Rich Transcription," in *Proc. of ICASSP*, 2005, vol. 1, pp. 205–208.