CONVERSATION-BASED LANGUAGE MODELING USING A LOSS-SENSITIVE PERCEPTRON ALGORITHM

Natasha Singh-Miller

MIT CSAIL
32 Vassar St., Cambridge, MA 02139
natashas@csail.mit.edu

Michael Collins

MIT CSAIL
32 Vassar St., Cambridge, MA 02139
mcollins@csail.mit.edu

ABSTRACT

Discriminative language models using n-gram features have been shown to be effective in reducing speech recognition word error rates (WER). In this paper we describe a method for incorporating discourse-level triggers and topic designations into a discriminative language model. Triggers are features identifying re-occurrence of words within a conversation. Topics represent clusters of related conversations. We introduce triggers that are specific to particular unigrams and bigrams, as well as "back off" trigger features that allow generalizations to be made across different unigrams. Topic-related features include unigram counts and features counting the number of topic related words in a sentence. We train our model using a new loss-sensitive variant of the perceptron algorithm that makes effective use of information from multiple hypotheses in an n-best list. We train and test on the Switchboard data set and show a 0.7 absolute reduction in WER over a baseline discriminative model which uses n-gram features alone, and a 1.7 absolute reduction in WER over the baseline recognizer.

***Index Terms*—** Perceptrons, Speech recognition, Natural languages

1. INTRODUCTION

Previous work on discriminative language modeling [1] has considered models where the optimal string $\mathbf{w}^*$ for a given acoustic input $\mathbf{a}$ is defined as follows:

$$\mathbf{w}^* = \arg\max_{\mathbf{w}} (\beta \log P_l(\mathbf{w}) + \log P_a(\mathbf{a}|\mathbf{w}) + \langle \bar{\alpha}, \Phi(\mathbf{a},\mathbf{w})\rangle)$$

In this approach, a standard language model, P_l, and an acoustic model, P_a, are used alongside a linear correction term $\langle \Phi(\mathbf{a},\mathbf{w}), \bar{\alpha}\rangle$.[1] $\Phi(\mathbf{a},\mathbf{w})$ is a *feature-vector representation* of the pair $(\mathbf{a},\mathbf{w})$, and $\bar{\alpha}$ is a parameter vector of the same dimensionality as $\Phi(\mathbf{a},\mathbf{w})$. The parameters $\bar{\alpha}$ are estimated using discriminative methods (e.g. the perceptron algorithm). Improvements in word error rate (WER) have been observed by incorporating both n-gram and syntactic features within $\Phi(\mathbf{a},\mathbf{w})$ [1, 2].

In this paper we consider three extensions to the discriminative language modeling approach. Our first contribution is to describe a method for including *trigger features* [3, 4] within the definition of $\Phi(\mathbf{a},\mathbf{w})$. Trigger features are designed to model the fact that content words are more likely to be used repeatedly within a single conversation than to occur evenly spread throughout all speech. For example the word "Uzbekistan" may occur very rarely, but within the context of a conversation where it has already occurred, the likelihood of seeing "Uzbekistan" again increases dramatically. To capture this phenomenon in our model, a trigger feature can be defined that indicates the number of times in a conversation that "Uzbekistan" is seen preceded by a previous instance of "Uzbekistan". In addition to lexically-specific trigger features, we also introduce *backoff* trigger features where content words are placed into different equivalence classes based on their TF-IDF scores [5]. The use of lexicalized trigger features within a generative language model, i.e., a model that attempts to estimate $P_l(\mathbf{w})$, is described in [3, 4]. However, our use of trigger features in a discriminative language model is arguably simpler and more direct—in particular, the parameter estimation method is more closely related to optimizing WER.

Additionally, we perform clustering to assign topics to each conversation which we then use to design new features for the discriminative model. The clustering method we use is hierarchical, clustering in a top-down fashion [6]. We add features for every unigram conditioned on a specific topic designation. We also include features indicating the number of topic words present in a given sentence, where a topic word is a word that occurs much more frequently in a given cluster than in the corpus as whole [7]. These features are introduced for several different levels of the topic hierarchy, from very general to very specific topic designations. Some previous approaches which utilize topic information to improve speech recognition performance include the design of maximum entropy models which use manually assigned topic labels to achieve a significant decrease in WER [7]. In [8] latent

[1] β is a positive constant that determines the relative weight of the language and acoustic models. We use $\langle x, y\rangle$ to denote the inner product of two vectors x and y.

semantic analysis is used to create a language model that displays lower perplexity on a large-vocabulary speech recognition task compared with an n-gram only language model. In another approach, language models are trained for separate topics, and at recognition time mixtures of these models are used along with caching to yield a significant reduction in perplexity [9]. Again our work differs from these previous approaches in the design of some of our features and because the discriminative model attempts to more directly optimize WER.

Our third contribution is to introduce a new loss-sensitive variant of the perceptron algorithm for the estimation of $\bar{\alpha}$. This perceptron is similar in form to that proposed by [10] for multiclass classification, however it explicitly models the loss of selecting different hypotheses, and also takes into account the fact that multiple hypotheses may be considered optimal. In contrast to the work in [1], this perceptron algorithm makes updates based on averaging the contribution from a larger number of hypotheses, potentially making much better use of the information in the hypothesis set.

We tested our model on the Switchboard corpus using the recognizer of [11] and the discriminative language model of [1] as baselines. Our model demonstrates a 0.7 absolute reduction in WER over the model in [1], and a 1.7 absolute reduction in WER over the baseline recognizer of [11].

2. FEATURES

In this section, we describe how to extend the discriminative model described above in order to include trigger and topic features. We will use the following definitions:

- $\mathbf{a}_1 \ldots \mathbf{a}_n$ represents a sequence of acoustic inputs constituting a single conversation.
- GEN($\mathbf{a}_i$) denotes the set of n-best hypotheses produced by the baseline recognizer for the acoustic input $\mathbf{a}_i$.
- $\mathbf{v}_i$ designates the transcription of $\mathbf{a}_i$ we use to construct histories for identifying triggering events.
- $\mathbf{h}_i = \{\mathbf{v}_1, \ldots, \mathbf{v}_{i-1}\}$ is the history of $\mathbf{a}_i$.
- $\Phi(\mathbf{a}_i, \mathbf{w}, \mathbf{h}_i)$ is a feature-vector representation. We assume that the score assigned by the generative model is the first feature in this vector (i.e., $\Phi_1(\mathbf{a}, \mathbf{w}, \mathbf{h}) = \log P_a(\mathbf{a}|\mathbf{w}) + \beta P_l(\mathbf{w})$).
- The resulting decoding model is:

$$\mathbf{w}_i^* = \arg\max_{\mathbf{w}} \langle \bar{\alpha}, \Phi(\mathbf{a}_i, \mathbf{w}, \mathbf{h}_i) \rangle$$

For training $\bar{\alpha}$, we assume that the baseline speech recognizer can be used to generate an n-best list of candidate hypotheses for any acoustic input. During training, $\mathbf{v}_i$ is the least errorful hypothesis in GEN($\mathbf{a}_i$). During decoding, $\mathbf{v}_i$ is the best scoring hypothesis under the generative model for each $\mathbf{a}_i$. We also experimented with defining $\mathbf{v}_i$ to be the hypotheses selected while decoding, but this gave neglible differences in performance.

The baseline discriminative model and our new model both include the following features. The first feature is the score assigned by the recognizer as described above. The remaining features include unigram, bigram, and trigram features. As one example, a trigram feature would be

$$\Phi_2(\mathbf{a}, \mathbf{w}, \mathbf{h}) = \text{\# of times \textit{the dog barked} appears in } \mathbf{w}$$

Similar features are defined for all unigrams, bigrams, and trigrams seen in the n-best lists of the training data.

2.1. Trigger Features

We augment the baseline model with trigger features designed to capture information about the re-occurrence of words. These features operate at the discourse level in that they depend upon the words of the current candidate hypothesis as well as all other words that have occurred in previous utterances in the conversation. The *unigram trigger* features, created for all unigrams seen in the training data, are of the following form.

$\Phi_3(\mathbf{a}, \mathbf{w}, \mathbf{h}) =$

1 iff: (a) *Uzbekistan* is seen in $\mathbf{w}$ at least twice; or (b) *Uzbekistan* is seen in $\mathbf{w}$ once and is seen at least once in the history $\mathbf{h}$

0 otherwise

In addition to unigram features, we include *bigram trigger* features. For example, we might have a feature that is similar to Φ_3 above, but tests for the bigram *San Francisco*. Features of this form are created for all bigrams seen in training data.

Since the above features are lexicalized—i.e., there is a separate feature for each distinct unigram or bigram—some may be very sparse within our training set. To counteract this shortcoming we introduce a set of *backoff trigger* features. Each word w in the vocabulary is assigned to one of eleven *bins* based on its TF-IDF score [5]. The TF-IDF score is defined as follows for any word w and conversation d, where w is seen in d:

$$score(w, d) = (1 + \log(tf_{wd}))(\log \frac{n}{df_w})$$

Here df_w is the number of conversations in which the word w occurs, n is the total number of conversations in training data, and tf_{wd} is the number of times word w occurs in conversation d. The score for a word w, which we will denote as $score(w)$, is the average of $score(w, d)$ over all conversations d that contain w. The function $score(w)$ attempts to measure the degree to which the word w is a content word (and thus is likely to be a good trigger feature). We calculated TF-IDF

Bin	Word Sample
bin_0	a, with, go
bin_1	down, rough, previous
bin_2	proportion, casual, fascinated
bin_3	junkie, michigan, orchid
bin_4	chagrin, docile, uniforms
bin_5	editorial, entrapment, salvation
bin_6	moderating, coronary, lures
bin_7	sonogram, pastrami, twix
bin_8	plentiful, infusion, smidgen
bin_9	theorists, viscous, sitar
bin_{10}	unethical, bisque, backswing

Table 1. A selection of a few words from each "backoff" trigger bin.
??

scores for each word seen in the training data, using the 4,800 transcribed conversation sides in the Switchboard training set as documents.[2]

Words are then placed into bins according to their score. Words with $score(w)$ less than 1.0 are assigned to bin_0. All remaining words are sorted by increasing score and divided into ten equal-sized bins. A few randomly chosen words from each bin are shown in Table **??**. In practice bin_0 consists of roughly the hundred most common words in speech (e.g. *a*, *with*, *go*, etc.). Since these words are so frequent, we anticipate that their trigger features will behave differently from the other words in the vocabulary. We create the other ten bins in this graded manner because we anticipate that different content levels will result in different trigger behavior.

One feature for each bin is then added to the model. Suppose Φ_w for any word w in the vocabulary is a trigger feature for that word (for example, $\Phi_{Uzbekistan}$ would be defined as in the example above). For each bin_b, for $b = 0 \ldots 10$, we define a feature as follows:

$$\Phi_b(\mathbf{a}, \mathbf{w}, \mathbf{h}) = \sum_{v \in bin_b} \Phi_v(\mathbf{a}, \mathbf{w}, \mathbf{h})$$

The feature Φ_b counts the number of triggering events involving the words in bin_b. These features allow the model to learn a general preference for triggering events involving each of the 11 bins.

2.2. Topic Modeling

We include topic information in the discriminative language model by classifying each conversation into a topic cluster and then introducing topic conditional features into the model. Like trigger features, topic features are designed to capture long distance relationships among words. By grouping together conversations of the same topic, we hope to learn how to adjust our language model to adapt to specific topics versus the whole corpus. For instance in a conversation about politics, words like "government" and "party" might be more prevalent than in the general case.

[2]Note that we used the reference transcriptions for calculating TF-IDF scores, as opposed to the outputs from the baseline recognizer.

2.2.1. Learning Topic Assignments

We learn topic assignments of the training data by performing unsupervised hierarchical clustering. To perform the clustering, we represent each conversation using a TF-IDF vector [5]. To be precise, each conversation d_i is represented as a vector v_i, where w_j is a word in the vocabulary $\mathcal{V}$ of size V. The function $score$ is the same TF-IDF function described previously.

$$v_i = [score(w_1, d_i), score(w_2, d_i), ..., score(w_V, d_i)]$$

We utilize a method known as bisecting k-means [6] to obtain a hierarchical clustering of the training conversations. The algorithm is a top-down clustering algorithm the operates as follows:

1. given vectors $v_1, ..., v_n$ and integer K where K is the desired depth of the clustering hierarchy

2. randomly select two vectors as initial means, and run k-means until convergence to obtain the initial two clusters

3. for $k = 2$ to K, split each cluster by randomly selecting two vectors in the cluster as initial means, and run k-means until convergence

This method of clustering partitions the training first into two clusters, then four clusters, then eight clusters, etc. In order to ensure that we don't have very small clusters, we do not bisect clusters containing fewer than 25 conversations. We can qualitatively test the effectiveness of the clustering by looking at the topic words associated with each cluster. The formal concept of a topic word that we use here was introduced by [7]. A topic word w of a cluster t is a word whose relative frequency in a cluster $f_t(w)$ is greater than its relative frequency througout the entire corpus $f(w)$. We identify topic words using the following topic word score.

$$f_t(w) \log \frac{f_t(w)}{f(w)}$$

Table 2 contains the twenty topic words with the highest topic word score from eight randomly chosen clusters after eight iterations of bisecting k-means. We can see from the topic words that the clustering algorithm seems to be learning clusters with a single coherent topic, such as space exploration in Cluster 193.

Cluster 193	**Cluster 399**	**Cluster 141**	**Cluster 396**
space	drug	fishing	gun
station	drugs	fish	guns
gravity	testing	boat	control
program	tested	caught	waiting
technology	test	trout	period
economy	privacy	bass	rifles
shuttle	positive	catch	weapons
research	random	lakes	hunting
circular	invasion	water	handguns
stations	tests	bottom	automatic
produce	company	coast	ban
limited	alcohol	catfish	shot
techniques	prescription	ocean	shotgun
funding	negative	salmon	criminals
effects	employees	fisher	purposes
mcdonald	drivers	fished	legitimate
effective	invaded	river	handgun
apollo	airline	lake	law
weightlessness	false	fresh	semiautomatic
exploration	drink	halibut	crime
Cluster 406	**Cluster 260**	**Cluster 307**	**Cluster 184**
school	daughter	paint	dinner
schools	children	painting	food
parents	mom	color	cheese
teach	beach	painted	party
children	trip	room	barbecue
education	fish	wallpaper	cook
grade	oldest	latex	gourmet
public	youngest	walls	cooking
teachers	girl	trim	turkey
child	family	white	mayonnaise
teacher	vacation	bathroom	hors
private	coast	dark	d'oeuvres
learning	girls	ceiling	chicken
system	indian	molding	sauce
kid	places	paints	serve
kindergarten	mother	built	recipe
plano	bus	hallway	salad
discipline	english	remodeled	meals
teaching	boy	finish	eat
elementary	golf	clean	ham

Table 2. Top twenty topic words of eight randomly chosen clusters after eight round of bisecting k-means clustering.

2.2.2. Assigning Topics to Test Conversations

In order to assign a new conversation d_{test} a topic, we first need to compute it's TF-IDF vector representation.

$$v_{test}(d) = [score_{test}(w_1, d_{test}), ..., score_{test}(w_{|V|}, d_{test})]$$

$$score_{test}(w, d) = (1 + \log(tf_{wd}))(\log \frac{n^{train}}{df_w^{train}})$$

Here df_w^{train} is the number of training conversations in which the word w occurs, n^{train} is the size of the training set, and tf_{wd} is the number of times word w occurs in conversation d. The TF-IDF vector representation of a test sample is computed using the training sample as the corpus for calculating the IDF portion of the score.

Once we have $v_{test}(d_{test})$, we simply assign d_{test} the topic of the cluster whose mean is closest to $v_{test}(d_{test})$. We can learn a topic assignment of d_{test} at each level of the cluster hierarchy, which we can use to attain either coarse or refined topic features.

2.2.3. Topic Features

We create the following features for each utterance hypothesis in a conversation assigned to a cluster k

- unigram features conditioned on topic designation
- three features, one indicating whether zero topic words, one topic word, or two or more topic words are present in a sentence conditioned on topic designation

These features are created for each topic in the hierarchy, so if there are eight hierarchical clusterings, features for each level are created for each utterance hypothesis. Note that at each level k of the hierarchy, $\frac{10000}{n(k)}$ topic words are chosen for each cluster, assuming $n(k)$ is the number of clusters at level k.

3. TRAINING: PERCEPTRON

Figure 1 show the loss-sensitive perceptron algorithm we use for training $\bar{\alpha}$. This perceptron is similar in form to the perceptrons proposed by [10] for multiclass classification and by [12] for reranking. The perceptron is loss-sensitive in two ways. First, the perceptron enforces a margin that scales linearly with increases in loss. Second, the perceptron recognizes that there may be multiple hypotheses with minimal loss that should all be considered optimal.

In a given n-best list, GEN($\mathbf{a}_i$), there may be one or more optimal hypotheses. For example, the correct transcription may not be present in the list, but there may be several hypotheses each with only one error, while all the other hypotheses have two or more errors. We denote the set of lowest error hypotheses of GEN($\mathbf{a}_i$) by G_i. In terms of performance, all members of G_i are considered optimal choices by the discriminative model.

Let B_i = GEN($\mathbf{a}_i$) $- G_i$, i.e. the set of all non-optimal hypotheses in GEN($\mathbf{a}_i$). Each hypothesis in B_i will display different numbers and types of errors. The following loss function is used to indicate the badness of each member of B_i:

$$\Delta_i(b) = edits(b) - edits(g) \text{ where } g \text{ is any member of } G_i$$

This loss function is simply the additional number of errors introduced by a hypothesis over the number of errors present in an optimal hypothesis. Note that all members of G_i have a loss of 0, while all members of B_i have a loss of 1 or greater.

We define a margin that scales as $\lambda\Delta_i(b)$ where $\lambda \geq 0$ is a parameter we select. Scaling the margin with the loss was originally proposed by [13], who give statistical bounds justifying this. Intuitively, the idea is to ensure that hypotheses with a large number of errors are more strongly separated from the members of G_i. In the experiments presented in this paper λ is always set to 1.0. We define the two sets $C_i \subseteq G_i$ and $E_i \subseteq B_i$ in Figure 1 which consist of optimal and non-optimal hypotheses, respectively, that violate the scaled margin. We then construct two new vectors $\sum_{c \in C_i} \tau(c)\Phi_i(c)$ and $\sum_{e \in E_i} \tau(e)\Phi_i(e)$, which are used to train the perceptron in the usual way. The values of τ must meet the constraints described in Figure 1. The first four constraints insure that the weights used to create the representative samples are all non-negative and sum to 1. The final constraint insures that the newly constructed average samples still violate the margin constraint in an averaged sense.

Note that the training examples used as input to the algorithm are constructed in the following way. $\mathbf{a}_1 \ldots \mathbf{a}_m$ is a sequence of acoustic representations formed by concatenating all conversations in the training data. The histories $\mathbf{h}_i$ are constructed as follows. We take $\mathbf{w}_i^*$ to be the member of G_i that is scored highest by the generative model. We define the history, $\mathbf{h}_i$, for utterance $\mathbf{a}_i$ to be the sequence $\mathbf{w}_{i-l}^*, \mathbf{w}_{i-l+1}^*, \ldots, \mathbf{w}_{i-1}^*$ where l is the number of previous utterances which belong in the current conversation.

There are many methods for selecting the values of τ. In this paper we consider the following simple definition:

$$\forall c \in C_i, \tau(c) = \frac{1}{|C_i|}$$

$$\forall e \in E_i, \tau(e) = \sum_{c \in C_i} \frac{v_c(e)}{|C_i| v_c^{total}}$$

$$v_c(e) = \begin{cases} 1 & \text{if } \langle \bar{\alpha}(\Phi_i(c) - \Phi_i(e)) \rangle < \lambda\Delta_i(e) \\ 0 & \text{otherwise} \end{cases}$$

$$v_c^{total} = \sum_{e \in E_i} v_c(e)$$

Essentially all the hypotheses in C_i receive an equal positive weight. The weights of the hypotheses in E_i are assigned based on the values $v_c(e)$. If $v_c(e)$ is 1 for many correct hypotheses c, $\tau(e)$ will be relatively high.

The more standard perceptron used in the baseline model can be thought of as a special case of this perceptron in which $\lambda = 0$ and the τ values are assigned as follows. We designate some $c' \in G_i$ as the single best hypothesis (for the baseline, the hypothesis in G_i with the best recognizer score). We update only if $c' \in C_i$. We set $\tau(c') = 1$ and $\tau(e) = 1$ where e is the member of E_i for which $\langle \bar{\alpha}, (\Phi_i(c') - \Phi_i(e)) \rangle$ is the lowest. All other τ values are set to 0.

Input: An integer T specifying the number of training iterations. A sequence of inputs $\mathbf{a}_1 \ldots \mathbf{a}_m$. A function GEN$(\mathbf{a}_i)$ that produces an n-best list of outputs for the input $\mathbf{a}_i$. A mapping $\mathbf{h}_i$ that represents the history of for $\mathbf{a}_i$. A function $\Delta_i(\mathbf{w})$ that represents the loss of selecting output $\mathbf{w}$ for the sample $\mathbf{a}_i$. Δ_i must always be non-negative and there must be at least one member of GEN$(\mathbf{a}_i)$ with a loss equal to 0.

Definitions: $G_i = \{\mathbf{w} | \mathbf{w} \in \text{GEN}(\mathbf{a}_i) \text{ and } \Delta_i(\mathbf{w}) = 0\}$
$B_i = \text{GEN}(\mathbf{a}_i) - G_i$
Let $\Phi_i(\mathbf{w})$ be shorthand for $\Phi(\mathbf{a}_i, \mathbf{w}, \mathbf{h}_i)$

Algorithm:
$\bar{\alpha} \leftarrow 0$ $\lambda \leftarrow 1.0$
For $t = 1$ to T, $i = 1$ to m

- $C_i = \{c | c \in G_i$ and $\exists z$ such that $z \in B_i$ and $\langle \bar{\alpha}, \Phi_i(c) - \Phi_i(z) \rangle < \lambda\Delta_i(z)\}$
- $E_i = \{e | e \in B_i$ and $\exists y$ such that $y \in G_i$ and $\langle \bar{\alpha}, \Phi_i(y) - \Phi_i(e) \rangle < \lambda\Delta_i(e)\}$
- If $|C_i| \neq 0$, define a function τ over $C_i \cup E_i$ such that the following constraints hold:
    * $\sum_{c \in C_i} \tau(c) = 1$
    * $\sum_{e \in E_i} \tau(e) = 1$
    * $\forall c \in C_i, \tau(c) \geq 0$
    * $\forall e \in E_i, \tau(e) \geq 0$
    * $\langle \bar{\alpha}, \left(\sum_{c \in C_i} \tau(c)\Phi_i(c) - \sum_{e \in E_i} \tau(e)\Phi_i(e)\right) \rangle < \lambda \left(\sum_{e \in E_i} \tau(e)\Delta_i(e)\right)$

  Update the parameters:
  $\bar{\alpha} \leftarrow \bar{\alpha} + \sum_{c \in C_i} \tau(c)\Phi_i(c) - \sum_{e \in E_i} \tau(e)\Phi_i(e)$

Output: The parameters $\bar{\alpha}$.

Fig. 1. The perceptron algorithm we propose for reranking speech recognition output. In our experiments we used the averaged parameters from the perceptron, see [1] for details.

We can prove some useful properties for the perceptron in Figure 1. Consider the case where the training data is linearly separable, or more specifically there exists some vector $\mathbf{U}$ and some maximal margin $\delta > 0$ such that $||\mathbf{U}|| = 1$ and the following constraint holds for all i:

$$\langle \mathbf{U}, (\Phi_i(g) - \Phi_i(b)) \rangle \geq \delta\Delta_i(b) \quad \forall b \in B_i, \forall g \in G_i$$

It can be shown that in a finite number of iterations, given that the values for τ satisfy the given constraints, the perceptron in Figure 1 learns a model $\bar{\alpha}$ that separates the data as follows:[3]

$$\langle \frac{\bar{\alpha}}{||\bar{\alpha}||}, (\Phi_i(g) - \Phi_i(b)) \rangle \geq \gamma \Delta_i(b) \quad \forall b \in B_i, \forall g \in G_i$$

where $\gamma = \frac{\lambda}{2\lambda + \frac{4R^2}{s}} \times \delta$, R is an upper bound on the maximum length of a sample feature vector, and s is the minimum size of the loss seen on an error. (For our loss function we have $s = 1$.) Note that as $\lambda \rightarrow \infty$, $\gamma \rightarrow \frac{\delta}{2}$.

4. EXPERIMENTS

We use the recognizer of [11] as our baseline recognizer (base-G) and to generate 1000-best lists used by the discriminative models. The discriminative model used in [1] also serves as a baseline (base-D). We train the rerankers using Switchboard [14], Switchboard Cellular [15], and CallHome [16] data. Rich Transcription 2002 (rt02) [17] data was used for development. Rich Transcription 2003 (rt03) [18] data was used for testing. The training set consisted of 5533 conversation sides (individual speakers in a conversation), or about 3.3 million words. The development set consisted of 120 conversation sides (6081 sentences) and the test set consisted of 144 conversation sides (9050 sentences).

The perceptrons train very quickly, usually converging within three passes over the training data, and we optimize the exact number of iterations using the development set. We report results for the test set only for the baselines and model that produces the best results on rt02.

We tested several combinations of the trigger features and report results in Table 3. We find that including all three types of trigger features–unigram self-triggers, bigram self-triggers, and backoff triggers–gives us the best results on the development set. This model gives us a 0.4% absolute reduction in WER over base-D and a 1.2% absolute reduction in WER over base-G on the development set. This optimal model also achieves a 0.5% absolute reduction in WER over base-D for the test set and a 1.5% absolute reduction in WER over base-G. The results on rt03 are significant with $p < 0.01$ using the sign test at the conversation level.

We also tested several combinations of the topic features as and report results in Table 3. We found that including multiple levels of topic features leads to decreased word error rates versus the use of just one level of topic features. Additionally we find that if the features of each level are indicator features as previous described but are scaled to $\frac{1}{3}$, the model performance improves. The value $\frac{1}{3}$ is used because three levels of topic features are used, and together the contribution of any unigram/topic combination sums to one. In this way the use of multiple hierarchical topic levels forms a sort of backoff scheme. We find that the topic-based discriminative model again gives us gives us a 0.4% absolute reduction in WER over base-D and a 1.2% absolute reduction in WER over base-G on the development set. This optimal model also

[3]For a proof of this see the appendix.

Features	**rt02**	**rt03**
Base-G	37.0	36.4
Base-D	36.2	35.4
Loss-sensitive perceptron: n-grams	36.0	35.3
+ unigram self-triggers	36.0	
+ bigram self-triggers	36.0	
+ backoff unigram self-triggers	35.9	
+ unigram and bigram self-triggers	35.8	
+ unigram and backoff unigram self-triggers	35.9	
+ unigram, bigram, and backoff unigram self-triggers	**35.8**	**34.9**
+ level 5 topic features	35.9	
+ level 2,4,6 topic features	35.9	
+ scaled level 2,4,6 topic features	**35.8**	**35.0**
+ unigram, bigram, backoff unigram self-triggers, scaled level 2,4,6 topic features	**35.7**	**34.7**

Table 3. Results of base-G, base-D, and our discriminative model on the development set (rt02) and the test set (rt03).

achieves a 0.4% absolute reduction in WER over base-D for the test set and a 1.4% absolute reduction in WER over base-G. The results on rt03 are significant with $p < 0.01$ using the sign test at the conversation level.

In the final model, consisting of a combination of the best-performing topic and trigger features, a 0.5% absolute reduction in WER is seen over base-D on the development set, and a 1.3% reduction in WER is seen over base-G. On the test set an absolute reduction in WER of 0.7% and 1.7% is seen over base-D and base-G respectively.

5. DISCUSSION

We created several bins for the backoff trigger features with the expectation that words with different frequencies and content levels would have different trigger behavior. The learned weights of these features for the final discriminative model are listed in Table 4. From bin_0 to bin_6 the learned weights increase. This confirms our hypothesis that words with increasing content levels (or bin numbers) are more influenced by triggering events. We see approximately a threefold increase in weight between bin_1 and bin_6, suggesting that the difference in behavior between the words in the two bins is quite large, and therefore it may be worthwhile to try to create a backoff scheme that is more sensitive to these differences. Finally, somewhat erratic weights are seen for bin_7 through bin_{10}. One reason for this may be that these are the rarest words in the training set, and therefore weights for these bins are not adequately trained.

The words which have the 40 highest weights for their associated unigram trigger features are listed in Table 5. These

bin_0	0.60	bin_4	13.51	bin_8	2.87
bin_1	6.82	bin_5	14.96	bin_9	15.34
bin_2	12.28	bin_6	18.66	bin_{10}	8.31
bin_3	13.65	bin_7	2.12		

Table 4. Weights of bin features in final model.

GONNA	WANNA	SOMEONE	LAKE
WEATHER	WOOD	ONE'S	TEND
TAUGHT	NORTH	OIL	SORT
WEAR	UH-HUH	LIVED	TRUCK
LIST	[LAUGH]	STAYED	ALRIGHT
ICE	NEWS	READING	SOMEPLACE
WRITE	AWHILE	COLOR	TOUGH
AGE	WONDERFUL	DINNER	CAUGHT
WOMEN	RIDE	SELL	POOL
MEAT	MEN	DEER	OUGHT

Table 5. The 40 unigram trigger features with highest weight in the final model.

include content words such as *truck* and *news*, as well as stylistic words such as *gonna* and *wanna*. We posit that words such as *gonna* get high trigger weights because they are more heavily used by some speakers than others. Interestingly, we see that many of the words in the list are homonyms of other words, such as *wear* and *where*, *wood* and *would*, *deer* and *dear*, and *weather* and *whether*. It seems likely that the occurrence of one of these words earlier in a discourse should make it more likely to see it later and help distinguish between homonyms.

In Table 6, the forty topic-specific unigram features with the highest learned weights in the final model are listed. Interestingly most of these features are not words at all but are interjection or breaths. This might be because these are very frequent in speech and may be present much more in certain styles of speech which have high correlation with topic designation. Many of these features had negative learned weights for other topic designations, supporting this hypothesis. Additionally, we again see short easily confusable words as with the trigger features.

The weights learned for the features counting the number of cluster specific topic words present in a sentence hypothesis are also interesting. The features indicating that no topic word is present always takes on a negative weight between -15.1 and -1.8 in the final model. The feature indicating that single topic word is present takes on a weight between -3.5 and 4.2, and the feature indicating the two or more topic words are present is always positive, with a learned weight between 1.5 and 15.0. These weights confirm our intuition that hypotheses are generally superior if they contain more topic words.

Finally we see that the perceptron algorithm we present

[BREATH]:125	UM-HUM:113	[BREATH]:30
ONE:75	SEEN:18	YOU:76
BECAUSE:125	WAS:75	[LAUGHTER]:81
UM-HUM:78	[BREATH]:76	READ:69
HER:114	HAVE:109	BUT:104
NOT:76	IS:122	UM-HUM:99
OR:79	UH-HUH:114	WEAR:79
HE:71	WAS:176	IS:76
WHERE:67	IT:84	LIKE:76
BUT:76	HIGH:5	DID:75
IF:29	HER:86	UM-HUM:121
UP:17	I-:28	UM:106
DIDN'T:71	HIM:71	SAYING:6
UH-HUH:75		

Table 6. The 40 unigram topic unigram features with highest weight in the final model. The number following the colon indicates the topic cluster associated with the feature.

provides additional gains over the baseline perceptron algorithm. Future work might consider alternative ways to select the parameters τ as this might lead to further gains.

6. PREVIOUS WORK

There is a large body of recent work on using discriminative methods to improve speech recognition performance [1, 2, 19, 20, 21, 22, 23, 24]. The work of [1, 2, 19] utilizes the same discriminative model used in this paper. While all three of these approaches train their model using the perceptron algorithm used as a baseline in this paper, [1] compares this technique with the use of conditional random fields (CRFs). The CRFs are trained by using the output of the perceptron algorithm to prune away features and initialize the parameters to be learned, and are able to achieve significantly better performance over the perceptron algorithm. Additionally, [1, 19] consider the use of word lattices and show little difference between training using lattices or 1000-best lists. Finally, in [2], the discriminative model is extended to use syntactic features which also significantly reduce word error rates.

Alternative decoding methods that optimize WER directly include the work of [20, 21, 22]. In [20] each member of an n-best list is considered an equally probable reference hypothesis and then during decoding a hypothesis is selected that minimizes the word error rate accordingly. In [21] this approach is extended to word lattices using a data structure called “word sausages” that allow decoding decisions to be made on a word by word basis. These structures are reused in [22] where decoding decisions are made using a set of learned rules. All these approaches display significant improvement to WER in large vocabulary speech recognition tasks.

In the work of [23, 24] methods of directly adjusting the parameters of a traditional language model are presented. In both, the parameters of a language model are adjusted to minimize the “loss” associated with selecting incorrect hypotheses over correct hypotheses on a training set. Both methods achieve significant reductions in WER for Chinese word recognition tasks. This approaches differs significantly from ours however as only the traditional language model parameters can be altered, whereas we have the freedom to define arbitrary features within our discriminative language model.

There is also an extensive body of work on creating adaptive language models for speech recognition [3, 4, 7, 8, 9, 25, 26, 27, 28]. In an early approach, [25] create an adaptive language model by creating a language model from recent context and interpolating this model with a static language model. Later approaches use caches, which store information on previously used n-grams and distance since last use, to adjust the language model as more context is revealed [9, 28]. Explicit triggers were incorporated into the language model by [3, 4] using maximum entropy techniques. Topic modeling was used by [7, 8, 9, 28, 29] with varied success. [9] explores the use of mixtures of topic-specific language models. [7] uses maximum entropy models and manually annotated topic labels to create a topic-adaptive language model. Finally, in the work of [8], the technique of latent semantic analysis is used to model long distance information. Most of these approaches achieve large drops in language model perplexity with less spectacular reductions in word error rate (if any).

Finally, there also is some interesting recent work on the perceptron algorithm. The work of [10], is most closely related to the algorithm we present here. They present an multiclass classification learning algorithm that similarly makes use of information about every violation and makes updates accordingly. In [12], similar reranking algorithms are presented as the one here with a few key differences. For example, here we construct a prototypical “good” and “bad” hypothesis for each sample and use these to update our parameters. In [12], every pair of hypotheses in an n-best list that violates the margin constraint leads to an independent update of the parameters. This potentially puts too much weight on long n-best lists and too little on shorter lists. Other noteworthy work on related online algorithms [30, 31] discuss alternative algorithm for learning full rankings as well as algorithms that weight the updates by the amount of loss generated by a single example.

7. CONCLUSION

In this paper we use the discriminative language model of [1] to create a reranker that includes discourse–level features. Specifically, we introduce trigger and topic based features that help the discriminative language model to adapt to discourse context. We use lexicalized and backoff trigger features that each show individual improvements and together make a substantial gain over the baseline model. Topic labels induced by hierarchical clustering also are shown to be effective at adjusting the language model for conversations with a specific topic. We see gains using lexicalized topic features and features that indicate the number of topic words present in a hypothesis. Additionally we present a perceptron algorithm used to train the discriminative model that shows improvements as well. Overall, the WER on the test set was reduced by 0.7 over the baseline discriminative model, and by 1.7 over the baseline recognizer. This work provides evidence that discriminative language modeling has the potential to deliver significant gains for speech recognition tasks. The success of the trigger features also shows how important discourse level information can be to transcribing spoken language.

8. REFERENCES

[1] B. Roark, M. Saraclar, M. Collins, and M. Johnson, “Discriminative language modeling with conditional random fields and the perceptron algorithm,” in *Proceedings of the 42nd Annual Meeting of the ACL*, 2004, pp. 48–55.

[2] M. Collins, B. Roark, and M. Saraclar, “Discriminative Syntactic Language Modeling for Speech Recognition,” in *Proceedings of the 43rd Annual Meeting of the ACL*, 2005, pp. 507–514.

[3] R. Lau, R. Rosenfeld, and S. Roukos, “Trigger-based language models: a maximum entropy approach,” in *proc. ICASSP-93*, 1993, vol. 2, pp. 45–58.

[4] R. Rosenfeld, *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*, Ph.D. thesis, Carnegie Mellon University, 1994.

[5] G. Salton and M.J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.

[6] M. Steinbach, G. Karypis, and V. Kumar, “A comparison of document clustering techniques,” in *KDD Workshop on Text Mining*, 2000.

[7] Sanjeev Khudanpur and Jun Wu, “Maximum entropy techniques for exploiting syntactic, semantic and collocational dependencies in language modeling,” *Computer Speech & Language*, vol. 14, pp. 355–372, 2000.

[8] J. R. Bellegarda, “Exploiting both local and global constraints for multispan statistical language modeling,” in *IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 1998, pp. 677–680.

[9] P. R. Clarkson and A. J. Robinson, “Language model adaptation using mixtures and an exponentially decaying cache,” in *Proc. ICASSP*, 1997, vol. 2, pp. 799–802.

[10] K. Crammer and Y. Singer, “Ultraconservative online algorithms for multiclass problems,” *Journal of Machine Learning Research*, vol. 3, no. 4-5, pp. 951–991, 2003.

[11] A. Ljolje, E. Bocchieri, M. Riley, B. Roark, M. Saraclar, and I. Shafran, "The AT&T 1xRT CTS System," in *Rich Transcription Workshop*, 2003.

[12] L. Shen and A. K. Joshi, "Ranking and reranking with perceptron," *Machine Learning*, vol. 60, pp. 73–96, 2005.

[13] B. Tasker, C. Guestrin, and D. Koller., "Max-margin markov networks," in *Neural Information Processing Systems Conference*, 2003.

[14] J.J. Godfrey and E. Holliman, "Switchboard-1 release 2," 1997.

[15] D. Graff, K. Walker, and D. Miller, "Switchboard cellular," 2001.

[16] A. Canavan, D. Graff, and G. Zipperlen, "Callhome american english speech," 1997.

[17] J.S. Garofolo, J. Fiscus, and A. Le, "2002 rich transcription broadcast news and conversational telephone speech," 2004.

[18] S. Strassel, C. Walker, and H. Lee, "Rt-03 mdetraining data speech," 2004.

[19] B. Roark, M. Saraclar, and M. Collins, "Corrective language modeling for large vocabulary ASR with the perceptron algorithm," in *Proc. ICASSP*, 2004, vol. 1, pp. 17–21.

[20] A. Stolcke, Y. Konig, and M. Weintraub, "Explicit word error minimization in n-best list rescoring," in *Proceedings of EUROSPEECH*, 1997, vol. 1, pp. 163–166.

[21] L. Mangu, E. Brill, and A. Stolcke, "Finding consensus in speech recognition: word error minimization and other applications of confusion networks," *Computer, Speech, and Language*, vol. 14(4), pp. 373–400, 2000.

[22] L. Mangu and M. Padmanabhan, "Error corrective mechanisms for speech recognition," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, 2001.

[23] Z. Chen, K.-F. Lee, and M.-J. Li, "Discriminative training on language model," in *Proceedings of ICSLP*, 2000.

[24] H. J. Kuo, E. Fosler-Lussier, H. Jiang, and C. Lee, "Discriminative training of language models for speech recognition," in *Proceedings of ICASSP*, 2002.

[25] F. Jelinek, B. Merialdo, S. Roukos, and M. Strauss, "A dynamic language model for speech recognition," in *Proceedings of the speech and natural language DARPA workshop*, 1991, pp. 293–295.

[26] D. Beeferman, A. Berger, and J. Lafferty, "A model of lexical attraction and repulsion," in *Proceedings of the 35th Annual Meeting of the ACL*, 1997, pp. 373–380.

[27] S. Della Pietra, V. Della Pietra, R. L. Mercer, and S. Roukos, "Adaptive language modeling using minimum discriminant estimation," in *HLT '91: Proceedings of the workshop on Speech and Natural Language*, Morristown, NJ, USA, 1992, pp. 103–106, Association for Computational Linguistics.

[28] R.M. Iyer and M. Ostendorf, "Modeling long distance dependence in language: topic mixtures versus dynamic cache models," in *IEEE Transactions on Speech and Audio Processing*, 1999, vol. 7, pp. 30–39.

[29] J. McDonough, K. Ng, P. Jeanrenaud, H. Gish, and J. R. Rohlicek, "Approaches to topic identification on the switchboard corpus," in *Proceesings of ICASSP*, 1994, pp. 385–388.

[30] K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer, "Online passive-aggressive algorithms," *Journal of Machine Learning Research*, vol. 7, pp. 551–585, 2006.

[31] K. Crammer and Y. Singer, "Pranking with ranking," *Advances in neural information processing systems*, vol. 1, pp. 641–648, 2002.

Appendix

Theorem: If there exists some vector $\mathbf{U}$ and some margin $\delta > 0$ such that $\|\mathbf{U}\| = 1$ and the following constraint holds for all examples $\mathbf{a}_i$ in training data

$$\langle \mathbf{U}, (\Phi_i(g) - \Phi_i(b))\rangle \geq \delta\Delta_i(b) \quad \forall b \in B_i, \forall g \in G_i$$

then in a finite number of iterations the perceptron in Figure 1 learns a model $\bar{\alpha}$ that separates the data as follows

$$\langle \frac{\bar{\alpha}}{||\bar{\alpha}||}, (\Phi_i(g) - \Phi_i(b))\rangle \geq \gamma\Delta_i(b) \quad \forall b \in B_i, \forall g \in G_i \quad (1)$$

where $\gamma = \frac{\lambda}{2\lambda + \frac{4R^2}{s}} \times \delta$. R is an upper bound on the maximum length of a sample feature vector ($\|\Phi_i(x)\|^2 \leq R^2$ for all $x \in \text{GEN}(\mathbf{a}_i)$). s is the minimum size of the loss seen on an error.

Proof: Let $l_k = \sum_{b \in E_i} \tau(b)\Delta_i(b)$, the loss on the kth mistake where $\mathbf{a}_i$ is the example leading to the that mistake. We know that

$$ks \leq \sum_k l_k \leq kr \quad (2)$$

where r is an upper bound on loss for any example in the training set.

We can now find an upper bound on $\|\bar{\alpha}_k\|^2$ where $\bar{\alpha}_k$ is the weight vector after the kth update.

$$\begin{aligned}\|\bar{\alpha}_k\|^2 =& \|\bar{\alpha}_{k-1} + \sum_{c\in C_i}\tau(c)\Phi_i(c) - \sum_{e\in E_i}\tau(e)\Phi_i(e)\|^2 \\ =& \|\bar{\alpha}_{k-1}\|^2 + \\ & 2\langle\bar{\alpha}_{k-1}, \sum_{c\in C_i}\tau(c)\Phi_i(c) - \sum_{e\in E_i}\tau(e)\Phi_i(e)\rangle + \\ & \|\sum_{c\in C_i}\tau(c)\Phi_i(c) - \sum_{e\in E_i}\tau(e)\Phi_i(e)\|^2 \\ \le& \|\bar{\alpha}_{k-1}\|^2 + 2\lambda l_k + 4R^2\end{aligned}$$

This step follows from the last constraint imposed by the algorithm on the definition of the function τ and the bound on the maximum length of a feature vector we defined above. Since $\bar{\alpha}_0$ is initialized to 0, we know that after k mistakes we have the following inequality.

$$\|\bar{\alpha}_k\|^2 \le 2\lambda\sum_k l_k + 4kR^2$$

Now using the inequality from Equation 2, specifically using the fact that $k \le \frac{\sum_k l_k}{s}$, we can derive the following inequality.

$$\|\bar{\alpha}_k\|^2 \le 2\lambda\sum_k l_k + \frac{4R^2\sum_k l_k}{s} \qquad (3)$$

We can now find a lower bound on the quantity $\langle\mathbf{U}, \bar{\alpha}_k\rangle$.

$$\langle\mathbf{U}, \bar{\alpha}_k\rangle = \langle\mathbf{U}, (\bar{\alpha}_{k-1} + \sum_{c\in C_i}\tau(c)\Phi_i(c) - \sum_{e\in E_i}\tau(e)\Phi_i(e))\rangle$$

The assumption made in Equation 1 about the separability of the training data can be applied to get the following inequality.

$$\langle\mathbf{U}, \bar{\alpha}_k\rangle \ge \langle\mathbf{U}, \bar{\alpha}_{k-1}\rangle + \delta l_k$$

Again, we know that after k mistakes we can get the following bound.

$$\langle\mathbf{U}, \bar{\alpha}_k\rangle \ge \delta\sum_k l_k \qquad (4)$$

We can use Equations 3 and 4 to find a lower bound on $\|\bar{\alpha}_k\|^2$ and use the bounds on $\|\bar{\alpha}_k\|^2$ to bound the total loss $\sum_k l_k$.

$$\begin{aligned}\|\bar{\alpha}_k\|^2 \ge& \langle\mathbf{U}, \bar{\alpha}_k\rangle^2 \\ 2\lambda\sum_k l_k + \frac{4R^2\sum_k l_k}{s} \ge& \delta^2(\sum_k l_k)^2 \\ 2\lambda + \frac{4R^2}{s} \ge& \delta^2(\sum_k l_k)\end{aligned}$$

$$\sum_k l_k \le \frac{2\lambda + \frac{4R^2}{s}}{\delta^2} \qquad (5)$$

The upper bound on the total loss is constant, and since we know that each mistake has a minimal loss of s, we can say that the algorithm must converge within a finite number of iterations. We can now calculate the final normalized margin achieved by the algorithm. Using Equation 3 and 5, we can derive the following inequality.

$$\begin{aligned}\|\bar{\alpha}_k\| \le& \sqrt{2\lambda\sum_k l_k + \frac{4R^2\sum_k l_k}{s}} \\ \|\bar{\alpha}_k\| \le& \sqrt{\left(2\lambda + \frac{4R^2}{s}\right)\left(\sum_k l_k\right)} \\ \|\bar{\alpha}_k\| \le& \sqrt{\left(2\lambda + \frac{4R^2}{s}\right)\left(\frac{2\lambda + \frac{4R^2}{s}}{\delta^2}\right)} \\ \|\bar{\alpha}_k\| \le& \left(2\lambda + \frac{4R^2}{s}\right)\frac{1}{\delta}\end{aligned}$$

Finally to calculate the normalized margin we take the final parameter $\bar{\alpha}_{final}$ which we know must adhere to the above boundary, and use it to normalize the margin achieved λ.

$$\frac{\lambda}{\|\bar{\alpha}_{final}\|} \ge \frac{\lambda}{(2\lambda + \frac{4R^2}{s})} \times \delta$$

The proof is complete.