Available online at www.sciencedirect.com
SCIENCE DIRECT®
Speech Communication 48 (2006) 276–287
SPEECH COMMUNICATION
www.elsevier.com/locate/specom

Utterance classification with discriminative language modeling ☆

Murat Saraçlar [a,*,1], Brian Roark [b,1]

[a] Department of Electrical and Electronic Engineering, Boğaziçi University, 34342 Bebek, Istanbul, Turkey
[b] Center for Spoken Language Understanding, OGI School of Science and Engineering at Oregon Health and Science University, 20000 NW Walker Rd., Beaverton, OR 97006, USA

Received 1 January 2005; received in revised form 17 June 2005; accepted 27 June 2005

Abstract

This paper investigates discriminative language modeling in a scenario with two kinds of observed errors: errors in ASR transcription and errors in utterance classification. We train joint language and class models either independently or simultaneously, under various parameter update conditions. On a large vocabulary customer service call-classification application, we show that simultaneous optimization of class, *n*-gram, and class/*n*-gram feature weights results in a significant WER reduction over a model using just *n*-gram features, while additionally significantly outperforming a deployed baseline in classification error rate. A range of parameter estimation approaches, based on either the perceptron algorithm or conditional log-linear models, for various feature sets are presented and evaluated. The resulting models are encoded as weighted finite-state automata, and are used by intersecting the model with word lattices.

Keywords: Speech recognition; Language modeling; Discriminative training; Utterance classification

1. Introduction

Discriminative modeling techniques, such as the perceptron algorithm and global conditional log-linear models, have been shown recently to provide significant word-error rate (WER) reductions over baseline system performance on Switchboard, using just *n*-gram count features (Roark et al., 2004a,b). Reductions in WER are critical for applications making use of automatic speech recognition (ASR), but the key objective of a particular application may be different. For example, the effectiveness of spoken document retrieval can be impacted by the accuracy of the underlying ASR system, but the system objective will

☆ Parts of this study were presented in a conference paper (Saraçlar and Roark, 2005).

* Corresponding author.
E-mail addresses: murat.saraclar@boun.edu.tr (M. Saraçlar), roark@cslu.ogi.edu (B. Roark).

[1] The authors were with AT&T Labs-Research when much of the work presented here was performed.

0167-6393/$ - see front matter © 2005 Elsevier B.V. All rights reserved.
doi:10.1016/j.specom.2005.06.010

ultimately be some sort of precision of retrieval metric. Classification of unrestricted customer utterances into a number of classes for interaction with an automated dialog system is another application that relies upon accurate ASR (Gorin et al., 1997), but the success of the application is often evaluated with respect to class-error rate (CER) not WER. If the ASR system is being optimized for use in such an application, discriminative training scenarios such as those cited above should be focused upon the system objective rather than WER.

Often, however, multiple objectives will be important to the application. For example, both WER and CER will be important to the application when named entities or other information is extracted from the ASR output, in addition to classification. Generally speaking, however, classifiers are optimized independently of the ASR models, either for the purpose of just returning the utterance class (e.g. Haffner et al., 2003), or for feeding a class or topic back into the language model (e.g. Wu and Khudanpur, 2002). ASR models are likewise rarely trained for classification (though see Riccardi and Gorin, 1998). Reduced WER, however, can improve classification, and improved classification can reduce WER, i.e. a joint model serves both objectives, and performance may be improved for both objectives through simultaneous optimization of the joint model parameters.

A discriminative language model of the sort described in (Roark et al., 2004a,b), with the objective of reduced error in transcription, can improve CER by virtue of providing better transcriptions to the classifier. Alternatively, these discriminative approaches can be straightforwardly extended to perform utterance classification in addition to lattice re-weighting, by adding possible class labels to the transcriptions and including class label features in the model. In this paper, we perform a range of experiments investigating the benefit of simultaneous optimization of parameters for both WER and CER reductions. We demonstrate that simultaneously optimizing parameter weights for both *n*-gram and class features provides significant reductions in both WER and CER.

Several papers have been published in recent years on discriminative techniques. For example, Stolcke et al. (2000) used an "anti-LM", estimated from weighted *N*-best hypotheses of a baseline ASR system, with a negative weight in combination with the baseline language model, and Kuo et al. (2002) used the generalized probabilistic descent algorithm to train relatively small language models which attempt to minimize string error rate on the DARPA Communicator task. Discriminative methods used for call classification include the generalized probabilistic descent algorithm (Kuo and Lee, 2003), corrective training (Cox, 2003), rational function growth transform (Chelba et al., 2003), rational kernels (Cortes et al., 2004), and large margin classifiers (Haffner, this issue).

The rest of the paper is structured as follows. First, we present the general approach we use for discriminative language modeling, following Roark et al. (2004a,b). Next, we describe how we add utterance class annotations to the training and extend the feature set to perform classification in addition to lattice re-weighting. Under this general approach, several possible parameter update conditions are described. In addition to using linear models, as previously reported in (Saraçlar and Roark, 2005), we explore the use of global conditional log-linear models. Finally, we perform an experimental evaluation of the range of approaches that have been presented, on a large-vocabulary customer service application.

2. Methods

Our approach is based on linear models that can be represented as weighted finite-state automata. The weights are estimated from training data in order to jointly minimize errors in transcription and classification.

2.1. Linear models for n-gram language modeling

We follow the linear modeling framework outlined in (Collins, 2002, 2004), and used for WER reduction in ASR in (Roark et al., 2004a,b). The approach allows us to learn a mapping from inputs $x \in \mathcal{X}$ to outputs $y \in \mathcal{Y}$. In the current case, $\mathcal{X}$ is a

set of utterances, and $\mathcal{Y}$ is a set of possible transcriptions and utterance classifications. The approach assumes:

- Training examples (x_i, y_i) for $i = 1, \ldots, N$, where y_i is the reference annotation of x_i.
- A function ***GEN*** which enumerates a set of candidates ***GEN***(x) for an input x, e.g. word-lattice paths with hypothesized classifications.
- A **representation** Φ mapping each $(x, y) \in \mathcal{X} \times \mathcal{Y}$ to a feature vector $\Phi(x, y) \in \mathbb{R}^d$.
- A **parameter vector** $\bar{\alpha} \in \mathbb{R}^d$.

The components ***GEN***, Φ and $\bar{\alpha}$ define a mapping from an input x to an output $F(x)$ through

$$F(x) = \underset{y \in \boldsymbol{GEN}(x)}{\operatorname{argmax}} \Phi(x, y) \cdot \bar{\alpha} \quad (1)$$

where $\Phi(x, y) \cdot \bar{\alpha}$ is the inner product $\sum_s \alpha_s \Phi_s(x, y)$. The learning task is to set the parameter values $\bar{\alpha}$ using the training examples as evidence. Note that in ASR, weights are negative log probabilities, which changes argmax to argmin in these algorithms.

There are many approaches to setting the parameters, $\bar{\alpha}$, given training examples (x_i, y_i). For WER reduction, Roark et al. (2004a) used the perceptron algorithm (Collins, 2002), and Roark et al. (2004b) used global conditional log-linear models, which are related to conditional random fields (Lafferty et al., 2001), with the same feature sets as Roark et al. (2004a). For this paper, we first use the perceptron algorithm, shown in Fig. 1. We use cross-validation on a held-out set to determine the number of iterations T; and at test time, the averaged perceptron parameter values were used to control for overtraining. See Roark et al. (2004a) for more details on this approach. The second approach we investigate is an extension of the global conditional log-linear models used in (Roark et al., 2004b).

Inputs: Training examples (x_i, y_i)
Initialization: Set $\bar{\alpha} = 0$
Algorithm:
For iteration $t = 1 \ldots T$, $i = 1 \ldots N$
Calculate $z_i = \operatorname{argmax}_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \bar{\alpha}$
If$(z_i \neq y_i)$ then $\bar{\alpha} = \bar{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, z_i)$
Output: Parameters $\bar{\alpha}$

Fig. 1. A variant of the perceptron algorithm.

2.2. Feature definitions and implementation

The feature set Φ investigated in the current paper includes:

(1) the scaled cost given by the baseline ASR system, i.e. $-\lambda \log P(A, W)$;
(2) unigram, bigram and trigram counts in the utterance, e.g. $C(w_1, w_2, w_3)$;
(3) the utterance class cl; and
(4) class-specific unigram and bigram counts, e.g. $C(cl, w_1, w_2)$.

Feature sets (1) and (2) are the same as those used in (Roark et al., 2004a,b), and their parameter weights can be efficiently represented in a deterministic weighted finite-state automaton (WFA), through the use of failure transitions (Allauzen et al., 2003a). See Roark et al. (2004a) for details in this efficient encoding, which is presented schematically in Fig. 2. Briefly, every state in the automaton represents an n-gram history h, e.g. $w_{i-2}w_{i-1}$, and there are transitions leaving the state for every word w_i such that the feature hw_i has a weight. There is also a failure transition leaving the state, labeled with some reserved symbol ϕ, which can only be traversed if the next symbol in the input does not label any transition leaving the state. This failure transition points to the

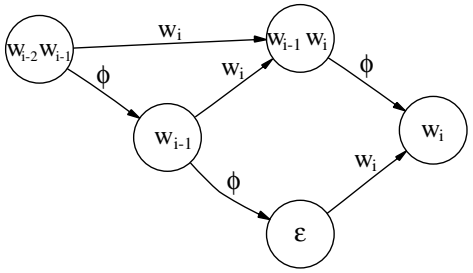

Fig. 2. Representation of a trigram model with failure transitions.

backoff state h', i.e. the n-gram history h minus its initial word. Let $\mathscr{N}$ denote this n-gram WFA.

The class-specific feature sets ((3) and (4) above) are encoded in a second, class-specific WFA, which we will denote $\mathscr{C}$. The initial state of $\mathscr{C}$ has k arcs, each labeled with one of the k class labels and weighted with the parameter weight for that class. The destination state of each of these k arcs is the start state of a class-specific n-gram automaton, of the same topology as $\mathscr{N}$.

The output of our system will be a class label and a transcription. However the weighted word lattice $\mathscr{L}$ output by the baseline recognizer only includes transcriptions, as does the class-independent model $\mathscr{N}$. We can emit a class label for every path from both of these automata by appending a new start state to the beginning of them. The new start state has k arcs, each labeled with one of the possible set of class labels and each has the original start state as destination. In such a way, any word string produced by the original WFA is preceded by any of the possible class labels. Let $\mathscr{L}_c$ and $\mathscr{N}_c$ denote $\mathscr{L}$ and $\mathscr{N}$ with class labels appended, respectively. Fig. 3 shows the start of such a trigram model $\mathscr{N}_c$, and Fig. 4 shows the start of the class-specific n-gram model $\mathscr{C}$.

The structure of the model in Fig. 4 is exactly the structure that would be used for a classifier based on generative n-gram models. In such an approach, a separate n-gram model would be estimated for each class, and the initial class-labeled arc would have a prior probability of the class, $P(C)$. The class $\widehat{C}(W)$ is then selected for word string W that maximizes the joint probability of class and word string W, i.e.

$$\begin{aligned}\widehat{C}(W) &= \operatorname*{argmax}_C \mathrm{P}(C, W) \\ &= \operatorname*{argmax}_C \mathrm{P}(C)\mathrm{P}(W|C) \end{aligned} \tag{2}$$

See Chelba et al. (2003) for a comparison of different parameter estimation techniques for classification based on n-grams.

Note that it is also possible to view this setup as a transduction from words to classes. Instead of using the output labels of a finite-state transducer, in our implementation we chose to use the initial labels to indicate the classes. We made this choice due to the fact that there is only one class label per utterance, as opposed to a sequence of labels (e.g. parts-of-speech). A single utterance initial symbol requires relatively little modification of the pre-existing approaches.

Using the joint model, the one-best class/transcript sequence is found by extracting the best path from $\lambda\mathscr{L}_c \circ \mathscr{N}_c \circ \mathscr{C}$, where $\circ$ denotes intersection as usual and λ is the scale given to the baseline ASR score. In contrast, most call routing systems first extract a one-best word transcript which is then used for classification. In our notation this corresponds to first extracting $w = \mathrm{bestpath}(\mathscr{L})$ and finding $\mathrm{bestpath}(w_c \circ \mathscr{N}_c \circ \mathscr{C})$, where w is the best word sequence and w_c is w with class labels appended. Examples of systems that use more than just the single best word hypothesis can be found in (Chelba et al., 2003; Cortes et al., 2004; Tur et al., 2004).

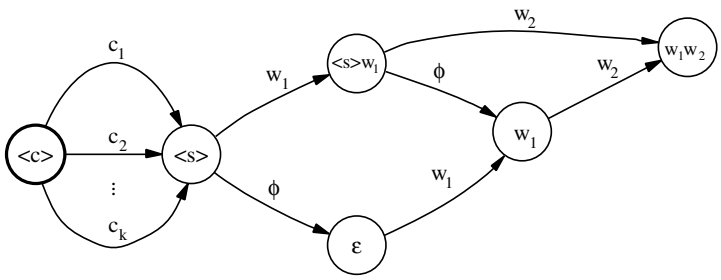

Fig. 3. Start of a trigram model with classes appended.

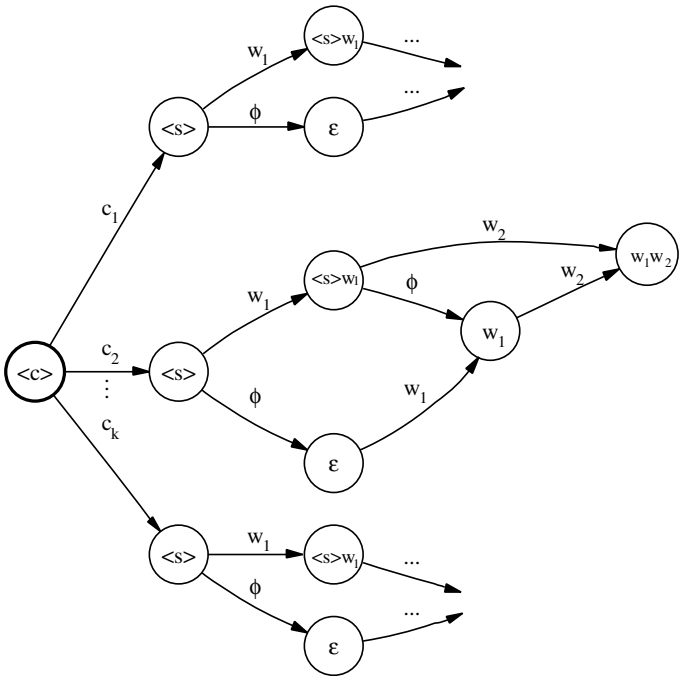

Fig. 4. Start of class-specific trigram model.

2.3. Parameter estimation

2.3.1. Perceptron algorithm

Perceptron training consists of extracting the best scoring annotation z for the current utterance x using the current models, and updating the parameter value for each feature with the difference between the gold standard feature count and the feature count of z. Which annotation to choose as the gold standard is an important question: in (Roark et al., 2004a) it was shown for n-gram modeling that using the minimum error rate annotation in the word-lattice as the gold standard outperforms using the reference annotation itself. This ability to choose the gold standard leads to two distinct ways of thinking about controlling the parameter estimation to meet a particular objective: either through feature selection or through gold standard selection. We can control parameter updates to suit a particular objective by choosing the relevant features; or by choosing the gold standard annotation so that only features related to that objective are updated.

Here we have two main objectives: having an accurate transcription and choosing the correct class for each utterance. The parameters can be independently or simultaneously optimized to achieve either one or both of the objectives. Let ***Gd*** be the correct class and minimum error rate word string for the utterance x, and ***1B*** the one-best annotation from $\lambda \mathscr{L}_c \circ \mathscr{N}_c \circ \mathscr{C}$ for the current models. Let $c(\boldsymbol{Gd})$ and $w(\boldsymbol{Gd})$ denote the class label and word transcription, respectively, of the annotation ***Gd***; and let $c(\boldsymbol{1B})$ and $w(\boldsymbol{1B})$ be defined similarly for the one-best output ***1B***. If the sole objective of training is classification error reduction, we can effectively ignore errors in transcription by setting our gold standard for parameter update to $y = c(\boldsymbol{Gd})w(\boldsymbol{1B})$. Since $w(y) = w(\boldsymbol{1B})$, the difference in feature values related to the baseline ASR score and class-independent n-gram counts will be zero, i.e. those parameters will not be updated, and will remain with value zero. Update will only occur if the class label $c(\boldsymbol{1B})$ is incorrect. Hence this effectively selects the features of the model to be from feature sets (3) and (4) from

Section 2.2, such that all bigram counts come only from the best scoring transcription. Similarly, if the sole objective is word-error rate, we can ignore the class label by setting $y = c(\boldsymbol{1B})w(\boldsymbol{Gd})$, which limits features to feature sets (1), (2) and (4), with class labels coming only from what is best scoring. Simultaneous optimization of the parameters occurs if we let $y = \boldsymbol{Gd}$. A variation can be obtained by optimizing the parameters for feature sets (1) and (2) for word error rate while optimizing the parameters for feature sets (3) and (4) for classification, i.e. n-gram counts come from only the best scoring hypotheses for features in set (4).

Note that in some cases one utterance may be annotated with more than one class. Similar to the uniform case of the algorithm described in (Crammer and Singer, 2003), we distribute the parameter updates uniformly over the classes. For example, if there are two classes labeled for an utterance, half of the reduction in cost of the class label feature goes to one class, and half goes to the other.

2.3.2. Global conditional log-linear models

Conditional log-linear models have been applied to NLP tasks such as parsing (Ratnaparkhi et al., 1994; Johnson et al., 1999), and tagging or segmentation tasks (Lafferty et al., 2001; Sha and Pereira, 2003; McCallum and Li, 2003; Pinto et al., 2003), as well as for n-gram language modeling for ASR (Roark et al., 2004b). These models use the parameters $\bar{\alpha}$ to define a conditional distribution over the members of $\boldsymbol{GEN}(x)$ for a given input x:

$$p_{\bar{\alpha}}(y|x) = \frac{1}{Z(x,\bar{\alpha})} \exp\left(\Phi(x,y)\cdot\bar{\alpha}\right)$$

where $Z(x,\bar{\alpha}) = \sum_{y\in\boldsymbol{GEN}(x)} \exp(\Phi(x,y)\cdot\bar{\alpha})$ is a normalization constant that depends on x and $\bar{\alpha}$. Note that the distribution is conditioned on the whole input, and not on the history. This differentiates *global* conditional log-linear models from history-conditioned exponential models whose parameters are determined using maximum entropy estimation.

Given these definitions, the log-likelihood of the training data under parameters $\bar{\alpha}$ is

$$\begin{aligned}\mathrm{LL}(\bar{\alpha}) &= \sum_{i=1}^{N} \log p_{\bar{\alpha}}(y_i|x_i) \\ &= \sum_{i=1}^{N} \left[\Phi(x_i,y_i)\cdot\bar{\alpha} - \log Z(x_i,\bar{\alpha})\right] \end{aligned} \quad (3)$$

Following Johnson et al. (1999) and Lafferty et al. (2001), we use a zero-mean Gaussian prior on the parameters resulting in the regularized objective function:

$$\mathrm{LL_R}(\bar{\alpha}) = \sum_{i=1}^{N} \left[\Phi(x_i,y_i)\cdot\bar{\alpha} - \log Z(x_i,\bar{\alpha})\right] - \frac{\|\bar{\alpha}\|^2}{2\sigma^2} \quad (4)$$

The value σ dictates the relative influence of the log-likelihood term vs. the prior, and is typically estimated using held-out data. The optimal parameters under this criterion are $\bar{\alpha}^* = \mathrm{argmax}_{\bar{\alpha}}\mathrm{LL_R}(\bar{\alpha})$.

As in (Roark et al., 2004b), we use a *limited memory variable metric* method (Benson and Moré, 2002) to optimize $\mathrm{LL_R}$. There is a general implementation of this method in the Tao/PETSc software libraries (Balay et al., 2002; Benson et al., 2002). This technique has been shown to be very effective in a variety of NLP tasks (Malouf, 2002; Wallach, 2002). The main interface between the optimizer and the training data is a procedure which takes a parameter vector $\bar{\alpha}$ as input, and in turn returns $\mathrm{LL_R}(\bar{\alpha})$ as well as the gradient of $\mathrm{LL_R}$ at $\bar{\alpha}$. The derivative of the objective function with respect to a parameter α_s at parameter values $\bar{\alpha}$ is

$$\frac{\partial \mathrm{LL_R}}{\partial \alpha_s} = \sum_{i=1}^{N}\left[\Phi_s(x_i,y_i) - \sum_{y\in\boldsymbol{GEN}(x_i)} p_{\bar{\alpha}}(y|x_i)\Phi_s(x_i,y)\right] - \frac{\alpha_s}{\sigma^2} \quad (5)$$

Note that $\mathrm{LL_R}(\bar{\alpha})$ is a convex function, so that there is a globally optimal solution and the optimization method will find it. The use of the Gaussian prior term $\|\bar{\alpha}\|^2/2\sigma^2$ in the objective function has been found to be useful in several NLP settings. It effectively ensures that there is a large penalty for parameter values in the model becoming too large—as such, it tends to control over-training. Having multiple σ^2 values for different feature sets

can yield further improvements (Chen and Rosenfeld, 1999). The choice of LL_R as an objective function can be justified as maximum a posteriori (MAP) training within a Bayesian approach. An alternative justification comes through a connection to support vector machines and other large margin approaches. SVM-based approaches use an optimization criterion that is closely related to LL_R—see Collins (2004) for more discussion.

One final implementation note, having to do with counting class-specific n-grams from the output of model composition. The GRM Library from AT&T (Allauzen et al., 2004, 2003b) contains all of the needed functionality to calculate the gradients for n-gram features, as detailed in (Roark et al., 2004b). To calculate the expected counts of n-gram sequences given a distribution represented by a word lattice, the function *grmcount* in the GRM Library composes the word lattice with a transducer of the structure shown in Fig. 5(a). The resulting transducer is then projected onto output labels, and epsilons are removed, yielding the correct expected counts for the n-gram sequences. If the initial label of each path in the lattice is a class label, we can produce class-specific n-gram counts by using a count transducer of the form in Fig. 5(b). This transducer consumes, and preserves, the initial class label, before performing the same function as the standard count transducer. With this small modification to the counting algorithm in the GRM library, we were able to efficiently calculate the gradients for this new set of features.

As in the perceptron case, this approach can be extended for the cases where one utterance is labeled with more than one class, i.e. when $c(\boldsymbol{Gd})$ is a set with more than one member. In this case, the probability of $y = c(y)w(y)$ given x is

$$p_{\bar{\alpha}}(y|x) = \frac{\sum_{c \in c(y)} \exp(\Phi(x, c, w(y)) \cdot \bar{\alpha})}{Z(x, \bar{\alpha})}$$

Denoting $y_i = c_i w_i$ the regularized objective function becomes

$$\mathrm{LL_R}(\bar{\alpha}) = \sum_{i=1}^{N} \left[\log \sum_{c \in c_i} [\exp(\Phi(x_i, c, w_i) \cdot \bar{\alpha})] - \log Z(x_i, \bar{\alpha}) \right] - \frac{\|\bar{\alpha}\|^2}{2\sigma^2}$$

whose derivative with respect to a parameter α_s at parameter values $\bar{\alpha}$ is given by

$$\frac{\partial \mathrm{LL_R}}{\partial \alpha_s} = \sum_{i=1}^{N} \left[\frac{\sum_{c \in c_i} \exp(\Phi(x_i, c, w_i) \cdot \bar{\alpha}) \Phi_s(x_i, c, w_i)}{\sum_{c \in c_i} \exp(\Phi(x_i, c, w_i) \cdot \bar{\alpha})} - \sum_{y \in \boldsymbol{GEN}(x_i)} p_{\bar{\alpha}}(y|x_i) \Phi_s(x_i, c(y), w(y)) \right] - \frac{\alpha_s}{\sigma^2}$$

As opposed to the perceptron case, there is an explicit objective function that is optimized. In addition to the joint optimization presented above, it is possible to explicitly optimize the parameters to achieve better utterance classification or word accuracy. Instead of using $\log p_{\bar{\alpha}}(y|x) = \log p_{\bar{\alpha}}(c, w|x)$ in the objective function, one could use $\log p_{\bar{\alpha}}(w|x) = \log \sum_c p_{\bar{\alpha}}(c, w|x)$ if word accuracy is the only goal or $\log p_{\bar{\alpha}}(c|x) = \log \sum_w p_{\bar{\alpha}}(c, w|x)$ if utterance classification is the only goal. It is also possible to approximate the summations by the best class or the best word string, respectively.

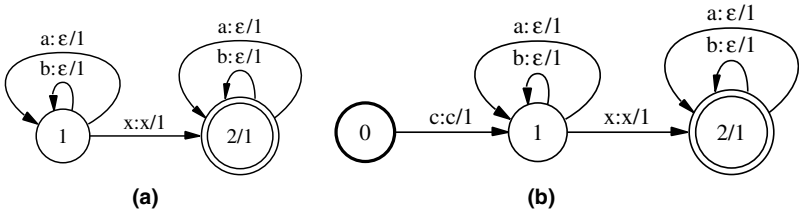

Fig. 5. Count transducers for (a) standard n-gram models; and (b) class-specific n-grams.

3. Experimental results

3.1. Experimental setup

We present experimental results on one of the AT&T VoiceTone® large vocabulary customer service applications (Gilbert et al., in press). The training set consists of 29,561 utterances (361,353 words), of which 5909 were used as held-out data. The test set consists of 5537 sentences (46,243 words). There are 97 utterance classes, referred to as *calltypes*. Each utterance in the corpus is labeled with one or more calltypes, e.g. `Request(Call_Transfer)` or `Request (Order_Status)`. On average there are 1.1 calltypes per utterance. There are at most four labels per utterance.

We evaluate performance with two metrics, corresponding to our two objectives. First is word accuracy (WAcc), which is standardly defined as 100-WER. Second is top-class error rate (TCErr), which is the percentage of utterances for which the highest scoring calltype is not among the labeled calltypes for that utterance. We followed two broad training scenarios with respect to the calltypes. In the first, we selected a single calltype per training utterance from among the set, always selecting that calltype which occurs least frequently in the training corpus. Selecting the calltype which occurs most frequently would make more sense for minimizing the TCErr. However, for most cases the secondary labels are very common and easy to learn labels such as 'yes' or 'no'. Our aim was to cover more labels and provide more training data for the less common labels. We refer to this training condition as "single-label". The second training condition leaves all calltypes, and parameter updates allocate evidence uniformly over the classes (similar to the uniform case of the algorithm described in (Crammer and Singer, 2003)).

3.2. Baseline results

The baseline deployed classifier (see Gupta et al., in press) for this application was trained using Boostexter (Schapire and Singer, 2000), using either reference transcriptions or one-best transcriptions from ASR. We trained a second baseline classifier with the perceptron algorithm by also restricting our input "lattice" to a single string. Classification is the sole objective here, since the word transcript is already fixed, so here we set the gold standard annotation to $y = c(\boldsymbol{Gd})w(\boldsymbol{1B})$. Table 1 shows some baseline results on this test data. For each classifier, we present three results. The first result is trained and tested on the one-best transcription of the baseline ASR system, which has a word accuracy of 78.4%. The second result is obtained with classifiers trained and tested on the one-best after intersecting the word-lattice $\mathscr{L}$ output from the baseline ASR with the perceptron-trained n-gram model $\mathscr{N}$, which improves word accuracy to 80.1%. The final baseline shows classification error when trained and tested on the reference transcription, as a lower-bound.

Table 1
Top-class error rate (TCErr) baselines using either the deployed classifier trained using Boostexter or a perceptron-trained classifier for single-label and multi-label training scenarios, given either (1) the one-best from the lattice $\mathscr{L}$ output from ASR; (2) the one-best after intersecting $\mathscr{L}$ with the corrective n-gram model $\mathscr{N}$; or (3) the reference transcription

Input word string (training and test)	WAcc	Boostexter	Perceptron	
		Multi-label TCErr	Single-label TCErr	Multi-label TCErr
Bestpath($\mathscr{L}$)	78.4	24.5	24.3	23.2
Bestpath($\mathscr{L} \circ \mathscr{N}$)	80.1	23.7	23.9	22.2
Reference	100.0	19.4	20.0	18.5

From this we can see that the perceptron classifier outperforms the Boostexter classifier by around 1% when trained on reference, but by more when trained on ASR output. This comparison is included primarily to demonstrate that the classifier that results from the baseline training algorithm is performing at a level comparable to other common approaches.

3.3. Experiments using linear models

Table 2 shows the results of training perceptron models on word-lattices under various parameter update conditions. Trials 1 and 2 show the result

Table 2
Word accuracy (WAcc) and top-class error rate (TCErr) for single-label and multi-label training scenarios under various parameter update conditions and with varying training input

Trial no.	Training and testing		Gold standard		Single-label		Multi-label	
	Classes appended	To lattice	Class label	Transcript	TCErr	WAcc	TCErr	WAcc
1	All classes	$\mathscr{L}$	$c(\boldsymbol{Gd})$	$w(\boldsymbol{1B})$	23.6	78.3	22.3	78.3
2	All classes	$\mathscr{L} \circ \mathscr{N}$	$c(\boldsymbol{Gd})$	$w(\boldsymbol{1B})$	23.2	80.2	21.6	80.3
3	True class	$\mathscr{L}$	$c(\boldsymbol{1B})$	$w(\boldsymbol{Gd})$	0.0	80.5	0.0	80.6
4	Trial 1 class	$\mathscr{L}$	$c(\boldsymbol{1B})$	$w(\boldsymbol{Gd})$	–	–	22.3	80.2
5	–	$\mathscr{L}_c \circ \mathscr{C}$	$c(\boldsymbol{1B})$	$w(\boldsymbol{Gd})$	–	–	35.5	80.5
6	All classes	$\mathscr{L}$	$c(\boldsymbol{Gd})$	$w(\boldsymbol{Gd})$	22.9	80.5	21.8	80.5

of setting the gold standard annotation to $y = c(\boldsymbol{Gd})w(\boldsymbol{1B})$, i.e. ignoring errors in transcription, similar to the baseline trials but without restricting the input to a single string. Trial 1 takes as input the baseline ASR lattice with all classes appended to the beginning of the lattice. In comparison with the baseline trial restricting input to the one-best from ASR, we see that we get nearly a 1% reduction in TCErr in the multi-label scenario by combining the models before performing one-best extraction. The word accuracy is not significantly different from the baseline. Trial 2 provides as input the word-lattice after intersection with the perceptron-trained n-gram model $\mathscr{N}$. Again, we observe a TCErr improvement (0.6% in the multi-label scenario), with a small non-significant change in word accuracy, this time slightly better than the baseline. Overall, this demonstrates the importance of training and applying these models to word-lattices rather than word-strings.

Trials 3–5 show the flip-side of trials 1 and 2, in that the gold standard annotation is chosen as $y = c(\boldsymbol{1B})w(\boldsymbol{Gd})$, i.e. calltype errors are ignored and only differences in word transcription are considered when updating parameters. Trial 3 provides an upper bound on the improvement to word accuracy that could be had from these calltype labels, since it appends the true reference class to each word lattice. Trial 4 demonstrates that, if instead of appending the true class, we instead append the predicted class, using the model trained in trial 1, we achieve no significant improvement in word accuracy. Note that the model used to provide the annotation was trained in a multi-label scenario, so this is the only scenario possible for trials 4 and 5. In trial 5, rather than restricting the label to a single predicted class, we simply take as input the word lattice intersected with the classifier from trial 1, which is analogous to trial 2 when the lattice was composed with the n-gram model $\mathscr{N}$. Here we find that the word error rate is improved almost to the level of the upper bound set in trial 3. However, the classification performance degrades significantly under this scenario. Finally, trial 6 performs simultaneous optimization of the class-independent and class-specific features by leaving the gold standard annotation as $\boldsymbol{Gd}$. Here we achieve significant word accuracy improvements over the perceptron n-gram model performance, as well as statistically indistinguishable TCErr performance from trial 2. For the multi-label case the word accuracy improvement (from 80.1% to 80.5%) is significant at $p < 0.005$ using the Matched Pair Sentence Segment Word Error significance test provided by SCTK (NIST, 2000). TCErr reductions from 23.2 to 21.8 or below are significant at $p < 0.05$.

We also experimented with optimizing the parameters for feature sets (1) and (2) for word error rate while optimizing the parameters for features sets (3) and (4) for classification. Table 3 shows the results for these trials, compared with using the same oracle for both feature sets. This approach, as expected, traded some word accuracy improvements for small classification improvements.

3.4. Experiments using global conditional log-linear models

Finally, we present preliminary results obtained by using global conditional log-linear models

Table 3
Word accuracy (WAcc) and top-class error rate (TCErr) for single-label and multi-label training scenarios with the same and different gold standards for different feature sets

Gold standard for feature sets		Single-label		Multi-label	
(1) and (2)	(3) and (4)	TCErr	WAcc	TCErr	WAcc
$c(\boldsymbol{Gd})w(\boldsymbol{Gd})$	$c(\boldsymbol{Gd})w(\boldsymbol{Gd})$	22.9	80.5	21.8	80.5
$c(\boldsymbol{1B})w(\boldsymbol{Gd})$	$c(\boldsymbol{Gd})w(\boldsymbol{1B})$	22.5	80.2	21.5	80.3

Table 4
Word accuracy (WAcc) and top-class error rate (TCErr) for the multi-label training scenario using the perceptron algorithm and global conditional log-linear (GCL) parameter estimation techniques

Parameter estimation	Multi-label	
	TCErr	WAcc
Perceptron algorithm	21.8	80.5
GCL, sequential optimization	21.2	80.9
GCL, conditioned on true class	0.0	81.7
GCL, simultaneous optimization	21.3	81.3

focusing on the multi-label scenario. Table 4 summarizes these results. First, we took the perceptron-trained word n-gram model $\mathcal{N}$ and re-estimated its parameters to optimize the regularized log-likelihood of the words. This improves the word accuracy to 80.9%. However, training a perceptron based classifier on the one-best word string we obtained a TCErr of 22.3%, no improvement over using the perceptron-trained word n-gram model $\mathcal{N}$. Further re-estimation of the classifier parameters to optimize the regularized log-likelihood of the classes given the one-best word string resulted in 21.2% TCErr. Next, we replicated trial 3 of Table 2. When true class labels are given it is possible to get 81.7% word accuracy by optimizing the regularized log-likelihood of the words given the true class labels. Simultaneous joint optimization yields improved results when using global conditional log-linear models.

Unfortunately, training is very slow in such an approach. The required summations are taken over the product space of possible classes and the word lattices. In fact, the full summation is not feasible for our task, so we had to prune the joint hypothesis space. Even so, the optimization process took about one CPU year before it was terminated. The relative speed of training recommends using either the perceptron algorithm or independent optimization of the parameter sets, even if this results in somewhat sub-optimal parameterizations. Of course, the speed of convergence is a function of the initial parameter values and the selected features. There might be better initializations that could make simultaneous optimization preferable.

4. Discussion

This paper has investigated the effect of joint discriminative modeling on two objectives, classification and transcription error. On the classification side, there are three potential factors leading to the best performance: (1) improvements in word accuracy provided by the model; and (2) delaying the extraction of the 1-best hypothesis, i.e. using word-lattice input rather than strings; and (3) simultaneously optimizing parameters for classification and transcription error reduction. In the multi-label scenario, the first two factors provide 1% reduction independently and 1.6% total reduction when combined. Simultaneous optimization does not provide further improvement over the two factors.

For word accuracy, a similar breakdown can be investigated, though with different conclusions. In this case, adding a predicted class to the word-lattice does not significantly improve word accuracy over the simple n-gram model. Providing the distribution over classes from the classifier, i.e. delaying the decision about which class is correct, provides a 0.4% absolute reduction in word error rate, though at the expense of a very large degradation of TCErr. Finally, simultaneous optimization of the parameters gives us as much improvement in word accuracy as we could get knowing the true class of the utterance, without penalizing TCErr. In summary, simultaneous optimization allows us to reach the best performance in both objectives with a single joint model.

Using global conditional log-linear models further improved the performance. Independently optimizing the parameters yielded a 0.5% improvement in word accuracy and a 0.8%

improvement in classification error over the perceptron algorithm.

For future directions, there are interesting issues when considering other annotations beyond utterance class. Coarser annotations, such as conversation topic in Switchboard, could be annotated, although because of topic drift and off-topic or topic-generic utterances, both the predictability and utility of these annotations may be less than in the current case. In addition, finer annotations, e.g. part-of-speech (POS) tags, bring up some difficult issues, particularly having to do with identifying an appropriate gold-standard, since manual annotation of a given training set would be expensive. In both of these cases, word accuracy is likely to be the primary objective of modeling, and from the current results, it seems clear that simultaneous joint modeling is a promising approach.

Acknowledgment

We would like to thank Gokhan Tur for help with the benchmark task data and the baseline used in our experiments.

References

Allauzen, C., Mohri, M., Roark, B., 2003a. Generalized algorithms for constructing language models. In: Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL), pp. 40–47.

Allauzen, C., Mohri, M., Roark, B., 2003b. GRM library. Available from: <http://www.research.att.com/sw/tools/grm/>.

Allauzen, C., Mohri, M., Roark, B., 2004. A general weighted grammar library. In: Proceedings of the Ninth International Conference on Implementation and Application of Automata.

Balay, S., Gropp, W.D., McInnes, L.C., Smith, B.F., 2002. Petsc Users Manual. Technical Report ANL-95/11-Revision 2.1.2, Argonne National Laboratory.

Benson, S.J., Moré, J.J., 2002. A limited memory variable metric method for bound constrained minimization. Preprint ANL/ACSP909-0901, Argonne National Laboratory.

Benson, S.J., McInnes, L.C., Moré, J.J., Sarich, J., 2002. Tao Users Manual. Technical Report ANL/MCS-TM-242-Revision 1.4, Argonne National Laboratory.

Chelba, C., Mahajan, M., Acero, A., 2003. Speech utterance classification. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Chen, S.F., Rosenfeld, R., 1999. A gaussian prior for smoothing maximum entropy models. Tech. Rep. CMU-CS-99-108, Carnegie Mellon University.

Collins, M., 2002. Discriminative training methods for hidden Markov models: theory and experiments with perceptron algorithms. In: Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP), pp. 1–8.

Collins, M., 2004. Parameter estimation for statistical parsing models: theory and practice of distribution-free methods. In: Bunt, H., Carroll, J., Satta, G. (Eds.), New Developments in Parsing Technology. Kluwer.

Cortes, C., Haffner, P., Mohri, M., 2004. Rational kernels: theory and algorithms. J. Mach. Learning Res. (JMLR) 5, 1035–1062.

Cox, S., 2003. Discriminative techniques in call routing. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Crammer, K., Singer, Y., 2003. A family of additive online algorithms for category ranking. J. Mach. Learning Res. 3, 1025–1058.

Gilbert, M., Wilpon, J.G., Stern, B., Fabbrizio, G.D., in press. Virtual agents for contact center automation. IEEE Speech Process. Mag.

Gorin, A.L., Riccardi, G., Wright, J.H., 1997. How may I help you? Speech Commun. 23, 113–127.

Gupta, N., Tur, G., Hakkani-Tür, D., Bangalore, S., Riccardi, G., Rahim, M., in press. The AT&T spoken language understanding system. IEEE Trans. Speech Audio Process.

Haffner, P., this issue. Scaling large margin classifiers for spoken language understanding. Speech Commun., doi:10.1016/j.specom.2005.06.008.

Haffner, P., Tur, G., Wright, J., 2003. Optimizing SVMs for complex call classification. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Johnson, M., Geman, S., Canon, S., Chi, Z., Riezler, S., 1999. Estimators for stochastic "unification-based" grammars. In: Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL), pp. 535–541.

Kuo, H., Lee, C., 2003. Discriminative training of natural language call routers. IEEE Trans. Speech Audio Process. 11 (1), 24–35.

Kuo, H.-K.J., Fosler-Lussier, E., Jiang, H., Lee, C.-H., 2002. Discriminative training of language models for speech recognition. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), Orlando, Florida.

Lafferty, J., McCallum, A., Pereira, F., 2001. Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proceedings of the 18th International Conference on Machine Learning, pp. 282–289.

Malouf, R., 2002. A comparison of algorithms for maximum entropy parameter estimation. In: Proceedings of the Sixth Conference on Natural Language Learning (CoNLL), pp. 49–55.

McCallum, A., Li, W., 2003. Early results for named entity recognition with conditional random fields, feature

induction and web-enhanced lexicons. In: Seventh Conference on Natural Language Learning (CoNLL).

NIST, 2000. Speech recognition scoring toolkit (SCTK) version 1.2c. Available from: <http://www.nist.gov/speech/tools>.

Pinto, D., McCallum, A., Wei, X., Croft, W.B., 2003. Table extraction using conditional random fields. In: Proceedings of the ACM SIGIR.

Ratnaparkhi, A., Roukos, S., Ward, R.T., 1994. A maximum entropy model for parsing. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP), pp. 803–806.

Riccardi, G., Gorin, A.L., 1998. Stochastic language models for speech recognition and understanding. In: Proceedings of the International Conference on Spoken Language Processing (ICSLP).

Roark, B., Saraclar, M., Collins, M., 2004a. Corrective language modeling for large vocabulary ASRwith the perceptron algorithm. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 749–752.

Roark, B., Saraclar, M., Collins, M., Johnson, M., 2004b. Discriminative language modeling with conditional random fields and the perceptron algorithm. In: Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL).

Saraçlar, M., Roark, B., 2005. Joint discriminative language modeling and utterance classification. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).

Schapire, R.E., Singer, Y., 2000. Boostexter: a boosting-based system for text categorization. Mach. Learning 39 (2/3), 135–168.

Sha, F., Pereira, F., 2003. Shallow parsing with conditional random fields. In: Proceedings of HLT-NAACL, Edmonton, Canada.

Stolcke, A., Bratt, H., Butzberger, J., Franco, H., Gadde, V.R.R., Plauche, M., Richey, C., Shriberg, E., Sonmez, K., Weng, F., Zheng, J., 2000. The SRI March 2000 Hub-5 conversational speech transcription system. In: Proceedings of the NIST Speech Transcription Workshop.

Tur, G., Hakkani-Tür, D., Riccardi, G., 2004. Extending boosting for call classification using word confusion networks. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 437–440.

Wallach, H., 2002. Efficient Training of Conditional Random Fields. Master's Thesis, University of Edinburgh.

Wu, J., Khudanpur, S., 2002. Building a topic-dependent maximum entropy language model for very large corpora. In: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP).