JOINT DISCRIMINATIVE LANGUAGE MODELING AND UTTERANCE CLASSIFICATION

Murat Saraclar[†] *and Brian Roark*[‡]

[†]AT&T Labs-Research, 180 Park Ave., Florham Park, NJ 07932, USA

[‡]Center for Spoken Language Understanding, OGI School of Science & Engineering at Oregon Health & Science University, 20000 NW Walker Rd., Beaverton, OR 97006, USA

[†]murat@research.att.com [‡]roark@cslu.ogi.edu

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. Using the perceptron algorithm, 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 update approaches for the 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.

1. INTRODUCTION

Discriminative modeling techniques, such as the perceptron algorithm and conditional random fields, have been shown recently to provide significant word-error rate (WER) reductions over baseline system performance on Switchboard, using just n-gram count features [12, 13]. 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 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 [7], 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. [8]), or for feeding a class or topic back into the language model (e.g. [16]). ASR models are likewise rarely trained for classification (though see [11]). Reduced WER, however, can improve classification, and improved classification can improve 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 [12, 13], 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 parameters weights for both n-gram and class features provides significant reductions in both WER and CER.

The rest of the paper is structured as follows. First, we will present the general approach that we are following for discriminative language modeling, following [12, 13]. Next, we will 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. Finally, we perform an empirical evaluation of the range of approaches that have been presented, on a large-vocabulary customer service application.

2. METHODS

2.1. Linear Models for N-gram Language Modeling

We follow the linear modeling framework outlined in [3, 4], and used for WER reduction in ASR in [12, 13]. 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 $\mathbf{GEN}$ which enumerates a set of candidates $\mathbf{GEN}(x)$ 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 $\mathbf{GEN}, \Phi$ and $\bar{\alpha}$ define a mapping from an input x to an output $F(x)$ through

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

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.

Inputs: Training examples (x_i, y_i)
Initialization: Set $\bar{\alpha} = 0$
Algorithm:
For iteration $t = 1 \ldots T, i = 1 \ldots N$
Calculate $z_i = \text{argmax}_{z \in \mathbf{GEN}(x_i)} \Phi(x_i, z) \cdot \bar{\alpha}$
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.

There are many approaches to setting the parameters, $\bar{\alpha}$, given training examples (x_i, y_i). For WER reduction, [12] used the perceptron algorithm [3], and [13] used conditional random fields [9] with the same feature sets as [12]. For this paper, we used the perceptron algorithm, shown in figure 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 [12] for more details on this approach.

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. $-\lambda \log \mathrm{P}(A, W)$;
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 [12, 13], and their parameter weights can be efficiently represented in a deterministic weighted finite-state automaton (WFA), through the use of failure transitions [1]. See [12] for details in this efficient encoding, which is presented schematically in Figure 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 backoff state h', i.e. the n-gram history h minus its initial word. Let $\mathcal{N}$ denote this n-gram WFA.

The class-specific feature sets (3 and 4 above) are encoded in a second, class-specific weighted WFA, which we will denote $\mathcal{C}$. The initial state of $\mathcal{C}$ 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 $\mathcal{N}$.

The output of our system will be a class label and a transcription. However the weighted word lattice $\mathcal{L}$ output by the baseline recognizer only includes transcriptions, as does the class-independent model $\mathcal{N}$. 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 $\mathcal{L}_c$ and $\mathcal{N}_c$ denote $\mathcal{L}$ and $\mathcal{N}$ with class labels appended, respectively. Figure 3 shows the start of such a trigram model $\mathcal{N}_c$.

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.

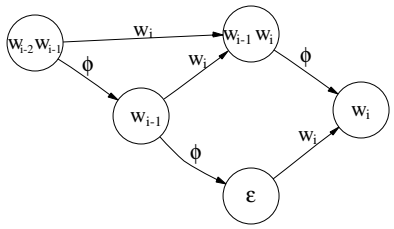

Fig. 2. Representation of a trigram model with failure transitions.

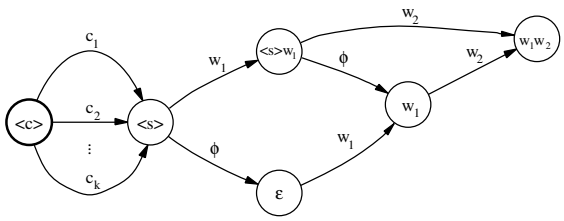

Fig. 3. Start of a trigram model with classes appended.

Using the joint model, the one-best class/transcript sequence is found by extracting the best path from $\lambda\mathcal{L}_c \circ \mathcal{N}_c \circ \mathcal{C}$, 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 $w = \text{bestpath}(\mathcal{L})$ and finding $\text{bestpath}(w_c \circ \mathcal{N}_c \circ \mathcal{C})$, 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 [2, 5, 15].

2.3. Parameter Estimation

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 [12] 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. We can control parameter updates to suit a particular objective 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 $\hat{y}$ be the gold standard annotation for utterance x, and z the one-best annotation from $\lambda\mathcal{L}_c \circ \mathcal{N}_c \circ \mathcal{C}$ for the current models. Let $c(\hat{y})$ and $w(\hat{y})$ denote the class label and word transcription, respectively, of the annotation $\hat{y}$; and let $c(z)$ and $w(z)$ be defined similarly for the one-best output z. 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 $y = c(\hat{y})w(z)$. Since $w(y) = w(z)$, 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 $c(z)$ is

incorrect. Similarly, if the sole objective is word-error rate, we can ignore the class label by setting $y = c(z)w(\hat{y})$. Simultaneous optimization of the parameters occurs if we let $y = \hat{y}$. A variation can be obtained by 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.

3. EMPIRICAL RESULTS

We present empirical results on one of the AT&T VoiceTone® large vocabulary customer service applications. The training set consists of 29561 utterances (361353 words), of which 5909 were used as held-out data. The test set consists of 5537 sentences (46243 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 4 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. We refer to this training condition as "1-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 [6]). 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.

The baseline deployed classifier for this application was trained using Boostexter [14], 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 $y = c(\hat{y})w(z)$. 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 percent. The second result is obtained with classifiers trained and tested on the one-best after intersecting the word-lattice $\mathcal{L}$ output from the baseline ASR with the perceptron-trained n-gram model $\mathcal{N}$, which improves word accuracy to 80.1 percent. The final baseline shows classification error when trained and tested on the reference transcription, as a lower-bound.

From this we can see that the perceptron classifier outperforms the Boostexter classifier by around one percent 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.

Table 2 shows the results of training perceptron models on word-lattices under various parameter update conditions. Trials 1 and 2 show the result of setting the gold standard annotation to $y = c(\hat{y})w(z)$, 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 percent reduction in TCErr in the k-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 $\mathcal{N}$. Again, we observe a TCErr improvement (0.6 percent in the k-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.

Input word string (training and test)	WAcc	Boostexter k-label TCErr	Perceptron 1-label TCErr	Perceptron k-label TCErr
bestpath($\mathcal{L}$)	78.4	24.5	24.3	23.2
bestpath($\mathcal{L} \circ \mathcal{N}$)	80.1	23.7	23.9	22.2
reference	100.0	19.4	20.0	18.5

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

Trials 3-5 show the flip-side of trials 1 and 2, in that the gold standard annotation is chosen as $y = c(z)w(\hat{y})$, 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 k-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, which is analogous to trial 2 when the lattice was composed with the n-gram model $\mathcal{N}$. 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 $\hat{y}$. 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 k-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[10]. 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. This variant yielded TCErr of 22.5 and 21.5, and WAcc of 80.2 and 80.3, for the 1-label and k-label cases respectively.

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

Trial	Training and testing input		Gold standard		1-label		k-label	
Number	Classes appended	To lattice	Class Label	Transcript	TCErr	WAcc	TCErr	WAcc
1	All classes	$\mathcal{L}$	$c(\hat{y})$	$w(z)$	23.6	78.3	22.3	78.3
2	All classes	$\mathcal{L} \circ \mathcal{N}$	$c(\hat{y})$	$w(z)$	23.2	80.2	21.6	80.3
3	True class	$\mathcal{L}$	$c(z)$	$w(\hat{y})$	0.0	80.5	0.0	80.6
4	Predicted class from trial 1	$\mathcal{L}$	$c(z)$	$w(\hat{y})$	-	-	22.3	80.2
5	-	$\mathcal{L}_c \circ \mathcal{C}$	$c(z)$	$w(\hat{y})$	-	-	35.5	80.5
6	All classes	$\mathcal{L}$	$c(\hat{y})$	$w(\hat{y})$	22.9	80.5	21.8	80.5

Table 2. Word Accuracy (WAcc) and Top-class error rate (TCErr) for 1-label and k-label training scenarios under various parameter update conditions and with varying training input.

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 k-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.

There are several important extensions to this work that are planned. First, it was shown in [13] that estimating the parameters of a discriminative n-gram model using conditional random fields significantly improves word accuracy over perceptron parameter estimation with the same feature set. The parameters of this joint model can also be estimated in this way, which may lead to improvements in one or both of the objective functions.

Second, 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.

Acknowledgments We would like to thank Gokhan Tur for help with the benchmark task data and the baseline used in our experiments. Much of the work reported here was done while the second author was at AT&T.

5. REFERENCES

[1] C. Allauzen, M. Mohri, and B. Roark. Generalized algorithms for constructing language models. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 40–47, 2003.

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

[3] M. Collins. 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)*, pages 1–8, 2002.

[4] M. Collins. Parameter estimation for statistical parsing models: Theory and practice of distribution-free methods. In H. Bunt, J. Carroll, and G. Satta, editors, *New Developments in Parsing Technology*. Kluwer, 2004.

[5] C. Cortes, P. Haffner, and M. Mohri. Rational kernels: Theory and algorithms. *Journal of Machine Learning Research (JMLR)*, 5:1035–1062, 2004.

[6] K. Crammer and Y. Singer. A family of additive online algorithms for category ranking. *The Journal of Machine Learning Research*, 3:1025–1058, 2003.

[7] A. L. Gorin, G. Riccardi, and J. H. Wright. How may I help you? *Speech Communication*, 23:113–127, 1997.

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

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

[10] NIST. Speech recognition scoring toolkit (SCTK) version 1.2c, 2000. Available at `http://www.nist.gov/speech/tools`.

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

[12] B. Roark, M. Saraclar, and M. Collins. Corrective language modeling for large vocabulary ASR with the perceptron algorithm. In *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 749–752, 2004.

[13] 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 Association for Computational Linguistics (ACL)*, 2004.

[14] R. E. Schapire and Y. Singer. Boostexter: A boosting-based system for text categorization. *Machine Learning*, 39(2/3):135–168, 2000.

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

[16] J. Wu and S. Khudanpur. 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)*, 2002.