CONTINUOUS SPACE DISCRIMINATIVE LANGUAGE MODELING

P.Xu[a], *S.Khudanpur*[a], *M.Lehr*[b], *E.Prud'hommeaux*[b], *N.Glenn*[d], *D.Karakos*[a], *B.Roark*[b], *K.Sagae*[c], *M.Saraçlar*[e], *I.Shafran*[b], *D.Bikel*[f], *C.Callison-Burch*[a], *Y.Cao*[a], *K.Hall*[f], *E.Hasler*[g], *P.Koehn*[g], *A.Lopez*[a], *M.Post*[a], *D.Riley*[h]

[a]JHU, [b]OHSU, [c]USC, [d]BYU, [e]Boğaziçi U., [f]Google, [g]Edinburgh, [h]Rochester

ABSTRACT

Discriminative language modeling is a structured classification problem. Log-linear models have been previously used to address this problem. In this paper, the standard dot-product feature representation used in log-linear models is replaced by a non-linear function parameterized by a neural network. Embeddings are learned for each word and features are extracted automatically through the use of convolutional layers. Experimental results show that as a stand-alone model the continuous space model yields significantly lower word error rate (1% absolute), while having a much more compact parameterization (60%-90% smaller). If the baseline scores are combined, our approach performs equally well.

Index Terms— Discriminative language modeling, neural network

1. INTRODUCTION

A language model (LM) assigns scores to word strings and plays an important role in automatic speech recognition (ASR) and many other applications. Under the predominant source-channel paradigm for speech recognition, the LM is generally used as the source of prior information, which evaluates the well-formedness of each hypothesis. Therefore, standard LMs are usually estimated from well-formed text in a maximum likelihood fashion. Specifically, the joint probability of a word sequence is often factorized into the product of local probabilities as below,

$$P(w_1, w_2, ..., w_l) = \prod_{i=1}^{l} P(w_i|h_i), \tag{1}$$

where h_i is the word history preceding the i^{th} word w_i. The set of conditional distributions $\{P(w|h)\}$ can be easily estimated based on empirical counts in the text corpus.

Most of the work presented here was done as part of a 2011 CLSP summer workshop project at Johns Hopkins University. We acknowledge the support of the sponsors of that workshop. This work was also partially supported by NSF Grants #IIS-0963898 and #IIS-0964102, and by TUBITAK (Project No: 109E142). Any opinions, findings, conclusions or recommendations expressed in this publication are those of the authors and do not necessarily reflect the views of the sponsors.

Discriminative training of LMs has been proposed as an effective complement to standard language modeling techniques [1, 2, 3]. Instead of attempting to learn a distribution over all possible word sequences, the discriminative objective directly targets the acoustically confusable hypotheses that the ASR system produces. To train such a model, we usually require transcribed speech data. An existing recognizer can be used to decode the speech and generate the corresponding confusion sets, and then discriminative training will learn to separate the lowest-error hypothesis from its set of competitors.

In discriminative language modeling, the LM is usually parameterized as a global linear model. Each word sequence W is associated with a score, which is the dot product between the feature vector $\Phi(W)$ and the parameter vector Θ. The probability of W is given by

$$P(W) = \frac{e^{\Phi(W)\cdot\Theta}}{Z}, \tag{2}$$

where Z is a normalizer. The features are in general represented symbolically, namely each word is treated as a distinct symbol. If n-gram features are used, each distinct word sequence of length up to n activates a different feature. This set of features can be very large and grows quickly with the training data.

In contrast, continuous feature representations are usually more succinct. Each word in the vocabulary is represented as a continuous vector. Word sequences can be represented as the concatenation of the representation for each word. Neural networks are powerful tools to learn such representations, they are also capable of learning non-linear features automatically. Therefore, it's not necessary to define any high order product features (e.g. bigrams, trigrams...) which often drastically increase the number of model parameters, all the complex non-linear interaction among words in the sequence can be discovered automatically.

Besides the space advantage, continuous feature representations are often believed to be able to generalize better. By mapping each word into a shared continuous space, word similarities and sequence similarities can be exploited, allowing for much more compact representations. We are able to fit a smaller number of parameters to a large training set, and potentially generalize better to unseen word sequences.

LMs have been trained in the continuous space before [4, 5], but only for the standard generative language modeling, where the goal is to model $P(w|h)$. This work is different in that it's the first *discriminative* LM trained in the continuous space. The most closely related work to our knowledge is [6]. The authors described a general deep architecture based on convolutional neural networks that can be used for various tasks. Our work resembles theirs in that we also use convolutional layers in our architecture, but the important distinction is that the task we have is a *structured* prediction problem, as opposed to more of the standard classification problems presented in their paper. Therefore, our proposed architecture carries noticeable differences.

The rest of the paper is organized as follows: We will present the training of standard discriminative LM in Section 2, the same loss function will be used for the continuous space model. We will then explain how to change the feature representation into continuous space in Section 3. Some training issues with the proposed model will be addressed in Section 4. Experimental results will be presented in Section 5.

2. PERCEPTRON AND MAXIMUM CONDITIONAL LIKELIHOOD TRAINING

The model in (2) can be trained in several ways. Perceptron training and maximum conditional likelihood (MCL) training are two commonly used approaches. Both methods share the similar intuition–increase the probability of the oracle hypothesis W^*, while penalizing other hypotheses in the set of likely candidates output by the recognizer. We denote this set as $GEN(A)$, where A stands for acoustics. The loss functions of the two methods are shown in (3) and (4),

$$\mathcal{L}_P = \left(-\log \frac{P(W^*)}{P(\hat{W})} \right)^+, \tag{3}$$

$$\mathcal{L}_{MCL} = -\log \frac{P(W^*)}{\sum_{W \in GEN(A)} P(W)}. \tag{4}$$

Perceptron training is an iterative procedure, which requires in each iteration identifying the best hypothesis $\hat{W}$ according the current model. The loss only occurs if the current model fails to rank the oracle hypothesis on top of the others. Meanwhile, MCL training directly considers all the competing hypotheses in $GEN(A)$, the loss exists for all training instances, therefore, the optimization is generally more complex.

It's worth noting that with the standard feature representation, it's generally hard to train with the MCL loss function without some kind of feature selection. The number of features in $GEN(A)$ is often too large to fit in memory. Fortunately, this problem will disappear with the continuous feature representation that we'll introduce in the next section.

3. FEATURE LEARNING USING CONVOLUTIONAL NEURAL NETS

Note that in (2), the score assigned to the word string is given by the dot product between the feature vector and the parameter vector. The model is linear in nature, and the features have to be specified beforehand. In order to capture the non-linear interactions among words in the sequence, we have to define features that consist of combinations of words (e.g. bigrams, trigrams...). The number of such features grows quickly with data. To describe our approach of representing features, we first replace the dot product with some non-linear function $g(W;\Theta)$, thus the model in (2) becomes

$$P(W) = \frac{e^{g(W;\Theta)}}{Z}. \tag{5}$$

The neural network can be used to parameterize $g(W;\Theta)$. It has the ability to learn proper embeddings for each word, and automatically extract non-linear features useful for our task. In order to assign a score to each word sequence W, our network architecture has to deal with input of variable length, which is not what the standard neural network can handle.

In [6], an architecture that can be built on top of variable-length word sequences was presented. Central to their methodology is the use of convolutional layers. Fig. 1 shows the first three layers of their architecture. Each word in the

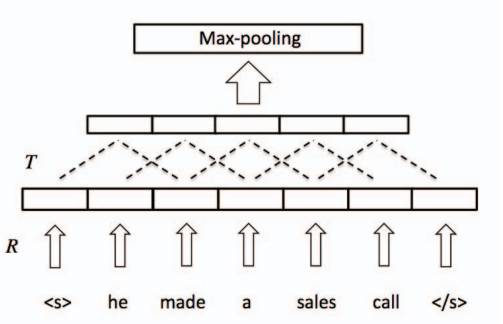

Fig. 1. Feature learning on word sequences proposed in [6]

sentence is associated with a continuous vector– this can be described as a look-up operation in the table R. The concatenated vector for the sentence does not have a fixed dimension either, therefore, features are extracted locally within fixed-size windows. As we can see, a linear transform T spans over only n words ($n = 3$ here) and is applied to every n-gram in the sentence. Non-linear functions such as $tanh$ are usually applied enabling nonlinear features to be detected.

For classification tasks such as the ones presented in [6], where the goal is to assign classes to the words in the input sequence, the features after the transform T in Fig. 1 usually have to go through a *max-pooling* layer which keeps only the top-k largest values in the feature vector. The resulting fixed-

size vector can be used as input to the standard neural network layers (e.g. softmax).

Discriminative language modeling is a different classification task in which the classes are word sequences. The features we extract from each individual sequence are not used to construct the distribution over classes directly, thus the max-pooling layer is not necessary. The neural net in our architecture only has to output a score $g(W)$ for each word sequence W. Therefore, the extracted features for the sequence can be summed up directly.

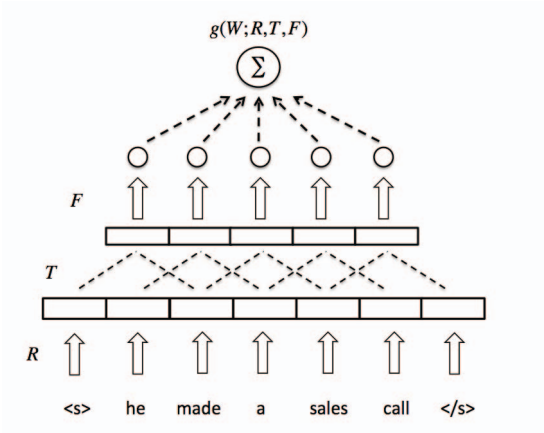

Fig. 2. Neural net representation of $g(W)$ used in this paper

Fig. 2 shows the proposed architecture to parameterize $g(W)$. The first two layers are the same as in Fig. 1. Before summing up the features, we apply another shared linear transform F, the result of which is a score assigned to the n-gram. Note that F can span over more than one feature vector in the previous layer. Such *stacking* of convolutional layers is often used for vision tasks [7], allowing for more global features to be extracted in upper layers. In order to obtain a score for the entire sequence, the scores for all n-grams are added up. Compared with the standard dot-product representation for $g(W)$, we replace the large number of n-gram features with a neural net structure that can be shared by all n-grams, leading to a much more compact model.

Once we have a score for every hypothesis in $GEN(A)$, we can easily compute $P(W)$ and train our model within the composed neural network. The complete architecture of our approach is shown in Fig. 3.

4. TRAINING ISSUES

Albeit large, the architecture in Fig. 3 is not difficult to train. Gradient descent methods can be easily applied to all parameters in R, T and F with the help of back propagation. Taking advantage of the shared structures and tied parameters, the optimization procedure can usually be greatly simplified.

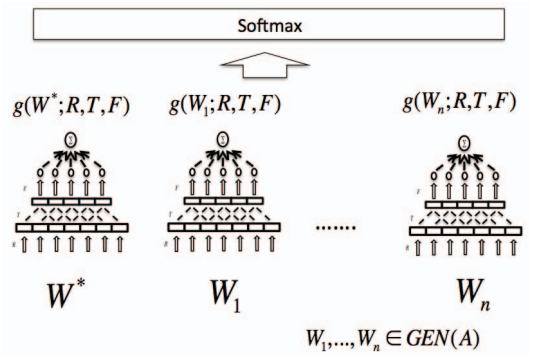

Fig. 3. Full architecture for discriminative LM

When computing $g(W)$, we have to go through all n-grams for each W in $GEN(A)$. Fortunately, $GEN(A)$ usually contains hypotheses that share a lot of n-grams in common. Therefore, computing scores for these n-grams only has to be done once for each training instance. The back propagation stage can also benefit greatly from the fact that n-gram scores are linearly combined. To illustrate this, write $g(W)$ as the sum of each n-gram scores, namely $g(W) = g(s_1^W) + g(s_2^W)... + g(s_{l_W}^W)$, where s_i^W denotes the i^{th} n-gram in W, and l_W is the total number of n-grams in W. Taking the gradient of the MCL loss function in (4), we have

$$\nabla \mathcal{L}_{MCL} = -\nabla g(s_1^{W^*}) - \nabla g(s_2^{W^*})... - \nabla g(s_{l_{W^*}}^{W^*}) + \sum_{W \in GEN(A)} P(W) \left(\nabla g(s_1^W) + \nabla g(s_2^W)... + \nabla g(s_{l_W}^W) \right). \quad (6)$$

As we can see, if some n-gram s appears the same number of times in W^* and all other W in $GEN(A)$, then $-\nabla g(s) + \sum_{W \in GEN(A)} P(W) \nabla g(s) = 0$, the contribution of s to the gradient is exactly zero. Therefore, it is also not necessary to go through all n-grams in the back propagation stage. Only n-grams that are not shared by $GEN(A)$ generate error signal to update the parameters.

In theory, any differentiable loss function can be applied on top of the architecture. However, the local optimum problem of training neural networks can make non-convex loss functions less desirable. The Perceptron training seems to have trouble reaching a good solution in our architecture. Therefore, we'll only report results using MCL trained models in the next section.

5. EXPERIMENTS & RESULTS

Our experiments are done on the English conversational telephone speech (CTS) dataset. The state-of-the-art IBM Attila recognizer is trained on 2000hrs of speech, half of them from the Fisher corpus, and the other half comes from Switchboard and Call-home. The baseline LM is a 4-gram LM trained on the 2000hrs of transcript containing about 25 million words,

plus close to 1 billion words of web data. For training the discriminative LM, the 2000hrs of transcribed speech data is divided into 20 partitions. The acoustic model trained on the 2000hrs is used to decode all the partitions. But the LM used to decode each partition only includes transcripts from the other 19 partitions. Such cross-validated language modeling is often used for training discriminative LM [2]. The hope is to avoid LM overfitting such that the confusions generated for each partition resemble those we'll see on the test data. We produce 100-best lists for all utterances in each partition for training discriminative LMs. The trained models are used to rerank the 100-best lists on the test data.

We train the standard averaged perceptron-based discriminative LM with trigram features in comparison with the continuous space discriminative LM (CDLM) using three training sets containing 2, 4, and 8 partitions respectively. The Dev04f corpus is used as the tuning set, consisting of 3.3hrs of speech. The final word error rates (WER) are reported on the 3.4hrs Eval04f corpus.

For training CDLMs, online gradient descent is used. The size of the feature extraction windows is fixed at three words, since it is generally thought that the standard discriminative LM generally does not benefit much from n-gram features beyond trigrams [2, 3]. The size of the word representation, and the size of the hidden layer are optimized on the development set. All the parameters in R, T, F are randomly initialized.

Table 1 shows the WER results of using the CDLM and the standard Perceptron DLM. Note that both the Perceptron and the CDLM scores can be combined with the baseline scores (AM+LM scores), which are generally important sources of information. The combination weights can be tuned on the develop set. As we can see, without the help of the baseline scores, the CDLM performs much better than the standard Perceptron. Nonetheless, the advantages disappear as the baseline scores are combined. The improvement achieved by the two approaches become almost identical. Being a stronger stand-alone model seems to prove the superior generalization ability enabled by the continuous feature representation. We may also have benefitted from the fact that the MCL loss function considers the entire confusion set, which is difficult to train with using the symbolic representation. However, such advantages are apparently offset by the acoustic model and the maximum likelihood trained LM.

Besides the competitive performance, our CDLM technique consistently produces much smaller models (ranging from 60% to 90% smaller), the sizes of which do not increase directly with more data. On the other hand, the number of features for the Perceptron DLM clearly has a tendency to increase as more training data becomes available.

		dev	eval	#parameters
1-best ASR		22.8	25.7	
Perceptron	2 parts	22.3/31.1	25.1/30.2	1.7M
	4 parts	21.8/29.5	25.1/29.3	3.3M
	8 parts	21.7/29.4	24.8/29.1	5.3M
CDLM	2 parts	22.3/29.7	25.2/29.0	0.74M
	4 parts	21.9/29.0	24.9/28.5	0.32M
	8 parts	21.6/28.5	24.7/28.0	1.8M

Table 1. WER of Perceptron DLM and CDLM. Each cell contains the WER with/without combining with the baseline scores.

6. CONCLUSION

We describe continuous feature representations for discriminative language modeling. Features are learned automatically through a novel neural network architecture.The resulting LM significantly outperforms the standard Perceptron DLM as a stand-alone model. When combined with the baseline scores, our model performs equally well. The proposed architecture also produces much more compact models, reducing the number of parameters by 60%-90% in our experiments.

7. REFERENCES

[1] H. Kuo, E. Fosler-Lussier, H. Jiang, and C. Lee, "Discriminative training of language models for speech recognition," in *Proc.of International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2002, vol. 1, pp. 325–328.

[2] B. Roark, M. Saraclar, and M.Collins, "Discriminative n-gram language modeling," *Computer Speech and Language*, vol. 21, pp. 373–392, 2007.

[3] Z. Li and S. Khudanpur, "Large-scale discriminative n-gram language models for statistical machine translation," in *Proc.of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA-2008)*, 2008.

[4] Y. Bengio, R. Ducharme, and P. Vincent, "A neural probabilistic language model," *Journal of Machine Learning Research*, vol. 3, pp. 1137–1155, 2003.

[5] H. Schwenk, "Continuous space language models," *Computer Speech and Language*, vol. 21, pp. 492–518, 2007.

[6] R. Collobert and J. Weston, "A unified architecture for natural language processing: deep neural networks with multitask learning," in *Proc.of ICML 2008*, 2008, pp. 160–167.

[7] Yann Lecun, Lon Bottou, Yoshua Bengio, and Patrick Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, 1998, pp. 2278–2324.