Deriving conversation-based features from unlabeled speech for discriminative language modeling

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

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

Abstract

The perceptron algorithm was used in [1] to estimate discriminative language models which correct errors in the output of ASR systems. In its simplest version, the algorithm simply increases the weight of n-gram features which appear in the correct (oracle) hypothesis and decreases the weight of n-gram features which appear in the 1-best hypothesis. In this paper, we show that the perceptron algorithm can be successfully used in a semi-supervised learning (SSL) framework, where limited amounts of labeled data are available. Our framework has some similarities to *graph-based label propagation* [2] in the sense that a graph is built based on proximity of unlabeled conversations, and then it is used to propagate confidences (in the form of features) to the labeled data, based on which perceptron trains a discriminative model. The novelty of our approach lies in the fact that the confidence "flows" from the unlabeled data to the labeled data, and not vice-versa, as is done traditionally in SSL. Experiments conducted at the 2011 CLSP Summer Workshop on the conversational telephone speech corpora Dev04f and Eval04f demonstrate the effectiveness of the proposed approach.

1. Introduction

The perceptron algorithm is a *supervised* learning algorithm, which takes as input labeled data and learns a model that separates the data that belong to different classes [3]. The perceptron has more recently been used successfully in the area of language modeling [1, 4], where it takes as input N-best lists from an ASR or MT system and returns feature weights that re-rank the N-best lists with the goal of scoring the less-erroneous hypotheses higher. The features used in [1, 4] are n-grams; these n-grams will necessarily have to have a non-negligible overlap between the training and the test data, to have some hope that the learned model will be able to generalize. To improve generalization, some additional features such as morphological, syntactic and trigger features have been investigated across a variety of languages and tasks [5, 6, 7, 8].

In this paper, we investigate the use of *unlabeled* audio and its derivatives (word lattices) in order to guide the generation of features. Despite the fact that the unlabeled audio cannot be used in the perceptron algorithm directly (by virtue of the fact that supervised transcriptions are missing), it can still provide some form of *well-formedness confidence estimates for the labeled data*. Therefore, even if the n-gram overlap is small, there are still features which provide additional means of confidence based on which the perceptron can learn to successfully re-rank the n-best lists. Although our proposal falls in the category of *semi-supervised learning* [9], it is different from the traditional notion of confidence (or label) propagation; our novelty lies in the fact that it is the *unlabeled* data which propagate their confidences to the labeled data, and not vice-versa. To the best of our knowledge, this is the first attempt at using unlabeled data to decide on the relevance or well-formedness of alternative hypotheses in the output of an ASR system.

Our experiments from conversational telephone speech (Dev04f and Eval04) show that the semi-supervised features offer statistically significant improvements over a strong ASR baseline, as well as a perceptron baseline which only uses 3-gram features.

The paper is organized as follows: Section 2 contains a description of the Perceptron algorithm: the original formulation by F. Rosenblatt, as well as its modified formulation for re-ranking problems in ASR and MT; Section 3 presents details about the semi-supervised features that are derived using unlabeled data; subsection 3.6 presents other features, including supervised language model features; Section 4 presents our experimental setup and word-error-rate (WER) results, and finally, Section 5 contains concluding remarks.

2. The Perceptron Algorithm

The perceptron algorithm was introduced by F. Rosenblatt in the 60s, for learning single-layer neural networks [3]. The basic idea is that, in the two-class separable case, one can learn an "optimum" feature weight vector simply by going through the training data and only updating the weight with those examples that get misclassified. In algorithmic terms, assuming that $\mathbf{w}(t)$ is the weight vector at time t, it is updated according to

$$\text{if } \mathbf{w}(t)\mathbf{x}_i y_i < 0, \text{ then } \mathbf{w}(t+1) = \mathbf{w}(t) + \eta \mathbf{x}_i y_i,$$

where η is the *learning rate*, and y_i is the label $(+1/-1)$ of the training example $\mathbf{x}_i$.

In the context of discriminative language modeling for ASR, following the approach of [10], [1] converted the problem of re-ranking the output hypotheses (e.g., an n-best list) into a multi-class problem: at each iteration of perceptron, the n-best list is reranked according to the current perceptron weight vector, and the weight vector is updated if the top-ranked hypothesis is not the lowest-error hypothesis.

The perceptron algorithm for discriminative language modeling is shown in Figure 1. Here, x_i represents the i-th utterance, GEN(x_i) is the output space (set of possible candidates generated for x_i by the ASR system), and y_i is the minimum-WER (oracle) hypothesis in GEN(x_i). (Alternatively, y_i could

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, as well as partial support by NSF #0963898 and #0964102. The research was done while the first author was with Johns Hopkins University.

be the manual transcription associated with x_i, but it turns out that using the oracle leads to a more stable algorithm.) Furthermore, $\Phi(x_i, z)$ is the set of features associated with hypothesis $z \in \text{GEN}(x_i)$.

Inputs: Training examples (x_i, y_i)
Algorithm: Set $\mathbf{w} = 0$
  For $t = 1 \ldots T$
    For $i = 1 \ldots N$
      Calculate $z_i = \arg\max_{z \in \text{GEN}(x_i)} \mathbf{w} \cdot \Phi(x_i, z)$
      If $z_i \neq y_i$, then update $\mathbf{w} = \mathbf{w} + \eta(\Phi(x_i, y_i) - \Phi(x_i, z))$
Output: Weight vector $\mathbf{w}$

Figure 1: The perceptron algorithm for discriminative language modeling used in [1], following [10].

In the approach taken in [1], only n-gram features were used. This forms our baseline discriminative language modeling condition. In the next section, we show how to augment Φ with a richer set of "semi-supervised" features, extracted using unlabeled data.

3. Features

In this section we describe the features that are derived using unlabeled data, along with a few others (derived directly from the n-best lists) that were used. In order to tackle the issue of limited training data, and to make them complementary to the baseline n-gram features used in perceptron, these "external" features should satisfy criteria such as (i) they should be limited in number, to avoid overfitting; (ii) they should correlate with (multiple, and hopefully diverse) notions of confidence. To show in what sense our proposal is similar to graph-based SSL, and to emphasize the differences, we briefly mention the setup of graph-based confidence propagation SSL.

3.1. SSL with Graph-based Label Propagation

In the standard setup of graph-based label propagation [2], the learner has a set of labeled data $(x_1, y_1), \ldots, (x_l, y_l)$ as well as unlabeled data $x_{l+1}, \ldots, x_n$ at its disposal. A graph is built over labeled and unlabeled data, based on some notion of proximity between data points. Next, "messages" are exchanged between the nodes of the graph, where neighbors of a node x encode their "belief" about the class label of x. An alternative view is to consider a random walk on the graph, which assigns label probabilities to nodes based on the probability that the walk will reach a node with one of the available labels. During this process, the label proportions of the labeled nodes always remain fixed, concentrated on the true labels [9].

Although we do generate graphs over the unlabeled data that connect them to the labeled data, what we propose is not the same as what is usually done in graph-based SSL [9]. Our approach is going in the opposite direction: we want to use the most relevant and confident unlabeled data to augment the features of the labeled data, over which discriminative training is applied. This is done by generating features over the n-best hypotheses of the labeled data. If a hypothesis is close to sufficiently many unlabeled data (or with sufficient strength) then there is an indication that the hypothesis is probably less erroneous than a hypothesis which is not close to any unlabeled data. Our assumption is that ASR errors occurring between different hypotheses are uncorrelated, and thus spurious cases of strong proximity to erroneous hypotheses are hopefully unlikely.

The "semi-supervised" features explored in this section encode various senses of proximity, relevance, or well-formedness: (a) Tf-idf similarity features: measure how close a hypothesis is to an unlabeled speech document; low proximity to *all* unlabeled speech documents is expected to correlate with bad-formedness. (b) Topic-model features: measures how relevant a hypothesis is to the topic distribution of the conversation it belongs to. The assumption is that correct hypotheses should be more likely than incorrect ones. (c) Clustering features: measures coherence according to some clustering of the unlabeled data. Combinations or words-clusters which appear correctly in the n-best lists get rewarded, while combinations of words-clusters which appear erroneously get penalized.

Details about these features are given in the rest of this section. It is assumed that the unlabeled data are decoded with the same recognizer used to decode the supervised data which train the perceptron, as well as any other development or test data. The decoded output consists of lattices (or confusion networks).

3.2. Quantized rank features

Before moving on to discussing the specific real-valued measures that we derive from the unlabeled data, we will first discuss our use of quantized rank indicators as the actual features being used in the discriminative language model. For each measure F, we score each hypothesis $\mathbf{h}$ in the n-best lists with $F(\mathbf{h})$ and rank the n-best list according to the score. The rank k is then used to define a boolean indicator feature, e.g., RANK.F=k. To avoid feature sparsity, we use quantized ranks, i.e., rank intervals. The intervals near the top of the list are shorter, to emphasize the higher ranks.

3.3. Tf-idf Similarity Rank Features

The lattices of the unlabeled data are used in order to compute a vector representation of each conversation side. In the information retrieval framework, a conversation side plays the role of a "document". Specifically, each conversation side is represented by a vector, whose i-th entry is set equal to a measure of how frequent the word is in the conversation side, discounted by a factor that measures its frequency in the whole collection. A similar representation can be used with other document definitions.

Since each tf-idf vector is compared (using cosine distance) to a single hypothesis, we follow [11] in using two definitions of term-frequency, in order to reduce the influence of very frequent words. In addition to typically defined term frequency, we also use the probability of occurrence of a word in a document. This is computed in a lattice by summing together the posterior probabilities of paths which contain the word, and can be done easily with FSM operations. These probabilities are then convolved together to give the probability of occurrence of a word in a document. This measure of frequency is always bounded above by 1.

Thus, we compute two *tf* quantities for word w in a document c; the first contains the expected count in c, while the second is the probability of occurrence in c:

$$tf_c^{(1)}(w) = \sum_{\text{utt} \in c} \sum_{n=1}^{\infty} n \Pr(w \text{ appears } n \text{ times in utt}), \quad (1)$$

and

$$tf_c^{(2)}(w) = \Pr(w \text{ appears} \geq 1 \text{ times in } c). \quad (2)$$

The document frequency (*df*) component is then calculated over the whole unlabeled corpus, generalizing directly to typical document frequency calculation:

$$df(w) \quad = \quad \sum_{c \in \mathcal{U}} tf_c^{(2)}(w). \quad (3)$$

The cosine similarity between a hypothesis $\mathbf{h} = (w_1, \ldots, w_k)$ and an unlabeled document $\mathbf{c} \in \mathcal{U}$ is then defined as

$$\text{sim}(\mathbf{h}, \mathbf{c}) = \frac{\langle \text{tfidf}(\mathbf{h}), \text{tfidf}(\mathbf{c}) \rangle}{\|\text{tfidf}(\mathbf{h})\| \cdot \|\text{tfidf}(\mathbf{c})\|} \quad (4)$$

where tfidf($\boldsymbol{h}$) is defined as a vector whose elements correspond to the words of the decoding vocabulary and are equal to $tf(w) \log(D/df(w))$, where D is the total number of documents in the unlabeled corpus. The notation $\langle \cdot, \cdot \rangle$ denotes vector inner product, and $\| \cdot \|$ denotes vector norm.

The features computed for each hypothesis $\mathbf{h}$ are the average and maximum of the set of tf-idf similarities:

$$\mathbf{f}^{(\text{AVG})}_{\text{tfidf}}(\mathbf{h}) = \frac{1}{D} \sum_{\mathbf{c} \in \mathcal{U}} \text{sim}(\mathbf{h}, \mathbf{c}), \quad (5)$$

$$\mathbf{f}^{(\text{MAX})}_{\text{tfidf}}(\mathbf{h}) = \max\{\text{sim}(\mathbf{h}, \mathbf{c})\}_{\mathbf{c} \in \mathcal{U}}. \quad (6)$$

This yields four features, since there are two versions of *tf* used to compute each of these features. We further double these to eight features by including versions of each that are multiplied by the ratio $|\mathbf{h}|/(|\mathbf{h}|+1)$, which is an increasing function of $|\mathbf{h}|$, as a means of penalizing very short hypotheses. All of these features are converted to boolean indicator features based on quantized ranks, as discussed in Section 3.2.

3.4. Topic-model Rank Features

The next set of features is computed using topic features. As described above, expected counts are computed from the lattices of the decoded audio and aggregated at the document (conversation side) level. These expected counts are then used to generate "pseudo-documents" as bags of words (each word is simply repeated as many times as the expected count, rounded to the nearest integer), based on which topic distributions are estimated with the Mallet toolkit [12]. Excluding function words, three different topic models are estimated, with 50, 100 and 200 topics respectively. Three more topic models are estimated (with 50, 100, 200 topics again), but with function words included.

Each topic t represents a distribution over words $p_t(w)$. To assign a topic-based likelihood to a hypothesis, all hypotheses of the conversation side it belongs to are first treated as a "document", based on which a distribution over topics $q^*(t)$ is estimated: $q^* = \arg\max_q \sum_{w \in \mathbf{c}} \log(\sum_t q(t) p_t(w))$.

Next, given this estimate of topic proportions, the log-likelihood of a hypothesis $\mathbf{h}$ is computed as

$$LL(\mathbf{h}) = \sum_{w \in \mathbf{h}} \log \left(\sum_t q^*(t) p_t(w) \right).$$

This log-likelihood is then used to rank hypotheses, and results in boolean indicator features based on quantized ranks, as discussed in Section 3.2. Six different log-likelihoods are used to produce features in this way, corresponding to the 3 models with different numbers of topics, each either with or without function words.

3.5. Clustering Features

The next set of features is based on clusters over the unlabeled audio documents. The pairwise similarity between all pairs of documents is first computed using eq. (4) as the similarity measure. This generates a weighted graph, which is subsequently thresholded such that only edges with a tf-idf similarity above θ are kept. The resulting graph is then given as input to a clustering algorithm [13] which optimizes the normalized-cut objective function

$$NCut(G) = \min_{C_1, \ldots, C_k} \sum_{i=1}^{k} \frac{\text{links}(C_i, \mathcal{U} \setminus C_i)}{\text{links}(C_i, V)},$$

where $\mathcal{U}$ is the set of vertices (conversation sides) in the unlabeled audio, C_i is the set of vertices of the i-th cluster, and the function links(A, B) is equal to the sum of the weights of the edges between vertices in A and vertices in B.

New clustered "documents" are now created: the number of words in each cluster is set equal to the sum of the expected counts (given by eq. (1)) of its constituent documents. Two values of θ (0.21 and 0.25) gave rise to two sets of features.

For each hypothesis $\mathbf{h}$, several symbolic features are generated as follows: for each word w in $\mathbf{h}$ whose *df* is in the lowest 95% percentile (that is, function words are largely excluded) all clusters $\mathcal{C}_w$ which contain the word are identified, and the cosine similarity (4) between $\mathbf{h}$ and each $C_i \in \mathcal{C}_w$ is computed. The symbolic features are then formed by concatenating together the identity of w and the identity of each cluster in $\mathcal{C}_w$ which results in the largest tf-idf similarity (to avoid sparsity, only the top 10% of the features are kept, over all $w \in \mathbf{h}$).

For example, if $\mathbf{h}$ contains the word *river*, then the generated features may look something like *(river,c36), (river,c41)*, etc.; this means that word *river* appears in clusters $c36, c41$, and these clusters are similar enough (according to tf-idf cosine similarity) to $\mathbf{h}$. What perceptron will do next during the training, will be to weight these features according to the correctness of the words, in the context of their associated cluster. That is, if *river* is wrong, then perceptron will learn that the features *(river,c36), (river,c41)* should be assigned a negative weight. Now, if *river* appears correctly in another utterance and is associated with another cluster (say, $c50$), then perceptron will increase its weight only in that context, without affecting what was learned for *(river,c36), (river,c41)*. This way, perceptron will capture the positive/negative correlation that exists between words and contexts they appear in, and thus improve the lexical coherence of each utterance.

3.6. Other Features

In addition to the semi-supervised features, other features derived from the n-best lists were used. Specifically:

- **length:** To make sure that hypotheses do not deviate too much from the "centroid" of the lengths in a n-best list, two "delta" features are computed for each hypothesis: the first is the rank of the absolute difference between the length of the hypothesis and the mean length in the n-best list, while the second uses the median length instead of the mean.
- **original rank:** The original rank of a hypothesis is also used as a separate feature.
- **LM features:** Nine language models, computed from a variety of sources, are used to compute additional "supervised" features. The nine sources were comprised of the supervised transcripts of all 2000 hours, and the web data that were obtained from UW [14]. In total, about one billion tokens were used to generate these language models. The likelihoods of the languages models (computed for all n-gram orders 1..5) were again converted to ranks.

All of these ranks were quantized as presented in Section 3.2.

4. Experimental Results

All our experiments were done on conversational telephone speech (CTS) corpora. A speech recognizer was trained on

2000 hours of speech using IBM's attila speech recognition software library [15]. The training data comprised roughly 1000 hours of conversations from the Fisher corpus, and another 1000 hours of conversations from the Switchboard and the Call Home corpora. The same 2000 hours were decoded with the learned acoustic model, but the decoding was done using 20-fold cross-validated language models. Specifically, for each fold i, all manual transcripts of the rest 19 folds (i.e., excluding fold i) were pulled together to build a 5-gram language model that was used to decode i.

Four folds, amounting to roughly 337 hours, were used as *labeled* data for training the various perceptron models. Another 15 folds, amounting to roughly 1545 hours, were used as *unlabeled* data for generating the semi-supervised features.

A small held-aside fold (not included in the above) was used as a stopping criterion for perceptron; the algorithm would stop if the WER did not improve on it for 5 consecutive iterations. The Dev04f corpus and the held-aside fold were used with cross-validation, for tuning the scaling factor used in the combination of the perceptron scores and the scores obtained by the baseline recognizer, as well as deciding which set of features to use on the test data. The Eval04f corpus was used as a blind test corpus for reporting WER results.

Table 1 shows the WERs on the dev & eval datasets, for three sets of features: (a) Plain 3-gram features (baseline perceptron); (b) 3-gram features and the best unsupervised feature (original rank) only; (c) the best semi-supervised feature set which does not contain the strong (supervised) language model features, tuned on the held-out and Dev04f data using cross-validation; (d) the best semi-supervised feature set which contains the supervised language model features (mentioned in Section 3.6), again tuned on the held-out and Dev04f data using cross-validation. As can be easily seen from these results, the semi-supervised features resulted in a significant improvement over the baseline on the eval data ($p < 0.01$). Inclusion of the language model features also resulted in a significant improvement over the baseline perceptron ($p < 0.05$).[1]

	dev	eval
1-best ASR	22.8	25.7
baseline percep (3gram)	21.9	25.1
percep (3gram+original rank)	21.8	25.0
percep (3gram+semi-sup features)	21.7	**24.8****
percep (3gram+semi-sup+LM features)	21.7	24.9*

Table 1: WER results using various feature sets.

5. Concluding Remarks

We have investigated the use of unlabeled audio for generating features that are given as input to a supervised discriminative algorithm for language modeling. These "semi-supervised" features attempt to convey various notions of confidence: relevance in terms of tf-idf cosine similarity, coherence as measured by topic models, and as measured by proximity to (unsupervised) clusters of the unlabeled audio. Results from conversational telephone speech transcription show that these features offer statistically significant WER gains, even surpassing the performance of strong language models trained on supervised data.

6. References

[1] B. Roark, M. Saraclar, and M. Collins, "Discriminative n-gram language modeling," *Computer Speech and Language*, vol. 21, no. 2, pp. 373 – 392, 2007. [Online]. Available: http://dx.doi.org/10.1016/j.csl.2006.06.006

[2] O. Chapelle, B. Schölkopf, and A. Zien, Eds., *Semi-Supervised Learning*. Cambridge, MA: MIT Press, 2006. [Online]. Available: http://www.kyb.tuebingen.mpg.de/ssl-book

[3] C. M. Bishop, *Pattern Recognition and Machine Learning*. Springer, 2006.

[4] 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.

[5] M. Collins, M. Saraçlar, and B. Roark, "Discriminative syntactic language modeling for speech recognition," in *Proc. ACL*, 2005, pp. 507–514.

[6] I. Shafran and K. Hall, "Corrective models for speech recognition of inflected languages," in *Proc. EMNLP*, 2006, pp. 390–398.

[7] N. Singh-Miller and M. Collins, "Trigger-based language modeling using a loss-sensitive perceptron algorithm," in *Proc. ICASSP*, 2007.

[8] E. Arisoy, M. Saraclar, B. Roark, and I. Shafran, "Discriminative language modeling with linguistic and statistically derived features," *IEEE Transactions on Audio, Speech and Language Processing*, vol. 20, no. 2, pp. 540–550, 2012.

[9] J. Bilmes and A. Subramanya, "Parallel graph-based semi-supervised learning," in *Scaling Up Machine Learning*, R. Bekkerman, M. Bilenko, and J. Langford, Eds. Cambridge University Press, 2011, forthcoming.

[10] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms," in *Proc. EMNLP*, 2002, pp. 1–8.

[11] D. Karakos, M. Dredze, K. Church, A. Jansen, and S. Khudanpur, "Estimating document frequencies in a speech corpus," in *Proc. of the 2011 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU-2011)*, 2011.

[12] A. K. McCallum, "MALLET: A machine learning for language toolkit," 2002, http://mallet.cs.umass.edu.

[13] I. Dhillon, Y. Guan, and B. Kulis, "Weighted graph cuts without eigenvectors: A multilevel approach," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 29, no. 11, pp. 1944–1957, 2007.

[14] I. Bulyko, M. Ostendorf, M. Siu, T. Ng, A. Stolcke, and O. Çetin, "Web resources for language modeling in conversational speech recognition," *ACM Trans. on Speech and Language Processing*, vol. 5, pp. 1:1–1:25, December 2007. [Online]. Available: http://doi.acm.org/10.1145/1322391.1322392

[15] S. F. Chen, B. Kingsbury, L. Mangu, D. Povey, H. Soltau, and G. Zweig, "Advances in speech transcription at IBM under the DARPA EARS program," *IEEE Trans. on Audio, Speech and Language Processing*, vol. 14, no. 5, 2006.

[1] We also did an ablation study where all (more than 1500) combinations of the aforementioned features were used on the held-aside and dev data. The combinations were ranked according to cross-validated WER, and each feature got a "score" equal to the reciprocal rank of the combination it appeared in. Based on this score, the top-ranked features were: original rank, clustering features (with $\theta = 0.25$), topic-model and tf-idf features.