DISCRIMINATIVE TRAINING OF LANGUAGE MODELS

Sridhar. Raghavan
Mississippi State University
raghavan@cavs.msstate.edu

ABSTRACT

This paper discusses the need for using discriminative training for language modeling and also discusses an approach to perform discriminative training on n-grams. The paper also explains the working principle of discriminative algorithms especially when applied to language modeling with the help of an example. The computational expense of doing discriminative training on language models is significantly high and hence researches around the world have come up with techniques to reduce the computational complexity. Some of the successful techniques use word lattices generated by an ASR system for discriminative training, and then update the language model probabilities on the word lattice so that they can be rescored easily. This paper reviews one such promising technique that estimates the parameters of a linear model using a variation of the well known perceptron algorithm. A WER reduction of 1.3% absolute on the switchboard task was reported by some researches using this algorithm.

1. INTRODUCTION

This paper describes a discriminative training technique that can be used for language modeling in an ASR system. State-of-the-art ASR systems use n-grams to model the language found in the training corpus N-grams have been proven to be successful in modeling word sequences in a language, but it suffers from the fact that it requires infinite amount of training data to optimally model all words (unigrams) or words sequences (n-grams) [1]. Also, another important thing to consider is that optimality in the model will not guarantee optimality in word error rate [1]. Hence it is not sufficient if the language model truly represents the underlying word distributions. For example an n-gram feature will try to separate likely sequences from unlikely sequences, but will not consider the actual confusability between the word pairs. In other words the relative score of the features is more important than the actual scores [1]. A discriminative algorithm tries to overcome the confusability issue by minimizing an error function, and this is done by iteratively correcting the parameters in the model. Error function can be some metric that is used to judge the performance of the system. There are several algorithms that could be used to find the optimum parameter estimates, and in this paper we will discuss one such algorithm known as the perceptron algorithm. The basic framework will be based on a linear model assumption and uses the perceptron algorithm for parameter estimation. The perceptron algorithm is used on word lattices and the final model obtained can be represented as a weighted finite state automaton (WFSA). The WFSAs are then used to readjust the language model weights in the word lattice. This technique gave an improvement of 1.3% absolute on the switchboard data [2]. Section 2 describes the need for discriminative training for language modeling and discusses an overview of discriminative training applied to speech recognition [2]. Section 3 describes the linear model framework that will be adopted for discriminative training of language models and will use the perceptron algorithm for parameter estimation [2], [3].

2. THE NEED FOR DISCRIMINATIVE TRAINING

A speech recognition system finds the most possible word sequence by using a combination of acoustic and language models. A statistical language model used in an ASR system is generally based on n-gram counts. N-grams are estimation of the word sequence distributions in a particular corpus. It has been experimentally observed that using higher order N-grams helps in decreasing the word-error-rate of an ASR system, but with several orders increase in the search space [4],[5], and in order to have a good estimate of the probability distribution of n-grams, one must require infinite amount of training data. The closeness of the language model to the application domain is obtained by computing the entropy, and the effectiveness of the language model can be indirectly observed by computing the entropy factor [5],[6]. But even with a well estimated language model we cannot guarantee optimum WER.

Since the WER is the main criterion, why not just optimize the models to decrease the WER instead of optimizing factors such as entropy? This was the main motivation behind discriminative training. An ASR system's framework can be exploited in a manner such that we can apply discriminative techniques to train the acoustic and language models For example the Bayes rule formulation inherently gives provision for discriminative training. Let "A" be the acoustic observations and "W" the

word sequence. The probability of the word sequence "W" given the acoustic vector "A" has to be maximized .i.e. P(W|A). By Bayes rule this can be written as:

$$P(W \mid A) = \frac{P(A \mid W)P(W)}{P_e(A)}. \quad (1)$$

The speech recognizer finds the optimal word sequence by maximizing the P(W) (language model) and P(A|W) (acoustic mode) probabilities which ultimately maximizes the posterior probability P(W|A). The denominator P(A) can be treated as a constant. This is called as maximum a posteriori criterion, but unfortunately this technique suffers from the fact that lack of training data will cause the models to converge to a sub-optimal estimate. This happens because the MAP tries to increase the probability of the correct model but fails to reduce the probability of the incorrect model. This is where we require discriminative training to remove the confusability between word pairs, and this is done by minimizing the denominator term P(A) instead of treating it as a constant. P(A) can be represented as in Equation 2.

$$P_e(A) = \sum_{i}^{M} P(A \mid W_i)P(W_i). \quad (2)$$

Computing the probability from Equation 2 is very expensive and also challenging, because training a single word sequence requires the system to consider all other possible sequences in the training set. Many algorithms have been proposed to ease the load of discriminative training. Determining all possible word sequences is a computationally intensive task, and we can use two techniques to do this: N-best list generation and word-lattice generation. Both these techniques are closely related, but it has been found that a word lattice is a much more representative of the true search space than the n-best list. Also, because the word lattices are represented in the form of an WFSA the ask of annotating language and acoustic model scores is done in a more efficient.manner In this paper we will see one such algorithm that is based on a linear model framework.

For recognition, relative scores between models are more important than absolute scores [1]. Hence a measure that determines the amount of confusability between word pairs is a very important parameter. In order to find the confusable words, we will have to first determine these words by doing a dynamic programming alignment of the output hypothesis with the gold standard word transcription. The misrecognition measure that specifies the relative difference between two confusable words can be represented as shown in Equation 3.

In Equation 3 "n" is a positive number and $d_i(A)$ is a distance measure which if greater than zero signifies misrecognition. This distance measure can be converted to a probability using a sigmoid function as shown in Equation 4.

$$d_i(A) = -P(A_i, W) + \left[\frac{1}{M^{-1}} \sum_{j, j \neq i} P(A_j, W)^n \right]^{1/n} \quad (3)$$

$$l_i(A) = l\big(d_i(A)\big) = \frac{1}{1 + \exp(-\gamma d_i(A) + \theta)}. \quad (4)$$

The sigmoid fit is the loss function which has to be minimized on the training set. Discriminative training is performed based on the recognition results obtained using a traditional language model which can be built using maximum likelihood estimation techniques. For every utterance in the training corpus the discriminative algorithm should increase the strength of correct word and weaken the incorrect words. Increasing or decreasing the strength of the N-grams is done by adding or negating counts respectively from the N-grams estimated in the original language model [1].

2.1. An example to demonstrate discriminative training on trigrams

Let the training utterance be *"The dog ate my little brother's pudding"* and the corresponding hypothesis through the recognizer is *"The dog hates my little brother's pudding"*. The errors between the two sentences is found by dynamic programming alignment, and in this example the word "*ate*" and "*hates*" are confusable. The correct word expected is "*ate*", and so the trigrams containing the word "*ate*" in the context as occurring in this sentence must be strengthened and the trigrams with the word "*hates*" in the context as occurring in the sentence must be weakened. This is achieved by adding a constant α to the correct trigrams and subtracting a constant β from the incorrect trigrams. The working of this simple algorithm is shown below and its influence on the model is shown in Figure 1:

C(the, dog, ate) = C(the, dog, ate) + α
C(the, dog, hates) = C(the, dog, hates) - β
C(dog, ate, my) = C(dog, ate, my) + α
C(dog, hates, my) = C(dog, hates, my) - β
C(ate, my, little) = C(ate, my, little) + α
C(hates, my, little) = C(hates, my, little) - β

In the above example C(.) is the n-gram count for the sequence. This type of discriminative training can help in reducing the confusability between word pairs. Some

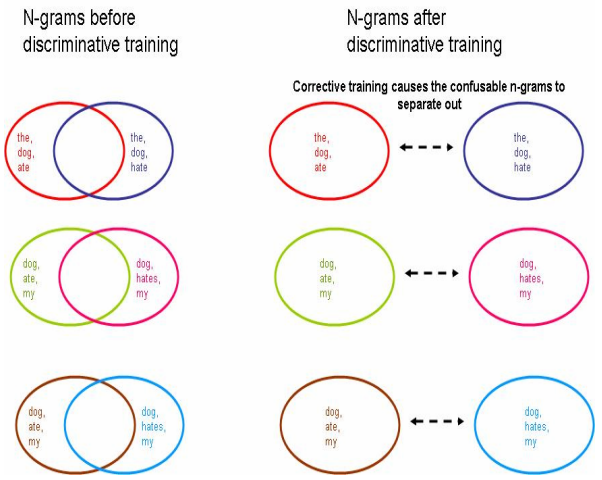

Figure 1 Effect of corrective training on n-grams, the figure demonstrates the effect on trigrams used in the example

authors have used the above technique to fine tune language models and have obtained about 5%-25% relative improvements in recognition error rates [1]. The drawback of an approach such as this one is that it is computationally very expensive and therefore cannot be practical for real applications. Recently many researchers have worked on using word lattices for discriminative training of language models, a word lattice from a speech recognizer would contain several possible word hypothesis for a particular utterance, and only one of them contain the optimal word sequence with the correct time marks, and hence the rest are used to weaken the less probable n-gram sequences present in the language model The following section will focus on using a word lattice for discriminative training of language models.

3. LINEAR MODEL FRAMEWORK AND PARAMETER ESTIMATION USING PERCEPTRON ALGORITHM

The linear model framework has been widely used in various tasks related to NLP [2]. The basic objective is to learn a functional mapping f(x) to relate an input 'x' to an output 'y'. In an ASR the input 'x' are the set of input utterances while 'y' is a set of all possible transcriptions for the given input utterance. The functional relationship f(x) is obtained as follows:

- Training examples (x_i, y_i) for i = 1…N
- A generator function GEN which generates all possible hypothesis of a given input 'x' GEN(x).
- A feature vector $\phi(x, y)$ which is a mapping of all (x, y) combination.
- A parameter vector $\bar{\alpha}$, which basically holds the weights that will be multiplied with the feature vector.

The functional mapping f(x) is obtained by using the above four parameters, and this is shown in Equation 5.

$$f(x) = \arg\max_{y \in GEN(x)} \Phi(x, y).\bar{\alpha} \; . \qquad (5)$$

Here the GEN(x) is a set of all possible candidate hypothesis generated for an utterance input "x". This can be approximated with a word graph generated by an ASR system. "y" will be a candidate sequence that maximizes the function f(x). $\Phi(x, y).\bar{\alpha}$ is an inner product of the parameter vector and the feature vector. The parameter vector $\bar{\alpha}$ is found by iterating over the training data. This is where the perceptron algorithm comes into picture.

The perceptron algorithm used for this task is a variant of the classic perceptron algorithm found by Rosenblatt [2], [8]. A perceptron is a simple form of feed forward neural network that can be used for linear classification. The perceptron is a linear classifier that maps the input to an output by a function f(x) that is iteratively computed. The parameters are the weights that are assigned to every feature element in the feature vector. The general formula of a perceptron algorithm is given in Equation 6

$$f(x) = < w, x > + b \; . \qquad (6)$$

Where "w" and "x" are the weight and input vectors respectively, and "b" is the bias. A simple perceptron model is shown in Figure 2

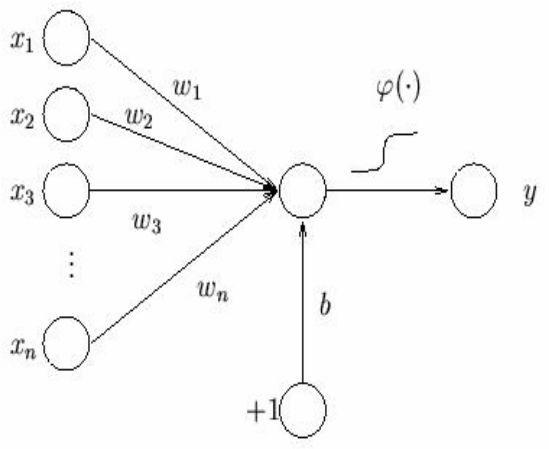

Figure 2 simple perceptron model

A variation of the conventional perceptron algorithm that will be used for language modeling is described below:

Input: Training examples (x_i, y_i)

Initialize the parameter vector $\bar{\alpha} = 0$

Algorithm:

for t = 1 to T, i= 1 to N

Calculate $z_i = \underset{z \in GEN(x_i)}{\arg\max} \Phi(x_i, z).\overline{\alpha}$

If $(z_i \neq y_i)$ then $\overline{\alpha} = \overline{\alpha} + \Phi(x_i, y_i) - \Phi(x_i, zi)$

Output is the parameter vector $\overline{\alpha}$

The perceptron algorithm converges only on data that is linearly separable. By using the perceptron algorithm we actually work under the framework described for discriminative training in section 2. If $z_i \neq y_i$ then the cost of the feature in z_i is increased by a constant and the cost of the feature in y_i is decreased by a constant. This type of corrective training will eventually helps in reducing the confusability among word pairs. Once the features on the model are updated the algorithm moves to the next utterance. After every pass on the training data, performance on the held out data set is evaluated. No further iterations are run if there is no improvement in the word error rate. The parameter set with the best performance is taken as the final model and will be used on the eval set.

The convergence criterion for the perceptron algorithm can be explained by considering the generating function GEN(x) [2] and the correct known transcription "y".

A function $\overline{GEN}(x_i)$ is defined such that $\overline{GEN}(x_i) = GEN(x_i) - \{y_i\}$ and the margin of separability is defined by a parameter δ, and this parameter is greater than zero if there are incorrect candidates in the hypothesis found in GEN(x). The parameter δ can be found by first determining a vector U such that ||U|| = 1. Hence the equation for finding the parameter δ is given in Equation 7:

$$U.\Phi(x_i, y_i) - U.\Phi(x_i, z) \geq \delta. \qquad (7)$$

The authors of the paper [2] have also found that the algorithm converges in just about one or two passes for the switchboard data. After every pass the weights and the corresponding features have to be updated. The features for n-grams are just the n-gram counts. Updating the original language model is done by constructing an WFSA for the discriminatively trained n-grams and then these are then used to update the language model weights in the word lattice. The updated word lattice is rescored for obtaining the error rate on the corrected language model [2].

A technique such as the perceptron algorithm can easily over train the language model and hence the performance on the test set will be poor, but it has been found that by averaging the parameter estimates obtained after each pass on every utterance, we can generalize the language model and hence obtain significant improvements on the test set.

4. CONCLUSIONS

Discriminative training on language model poses a lot of challenges when it comes to computational feasibility, even on the powerful modern day computers. In this paper we reviewed the discriminative approach applied to language modeling. We also saw the use of discriminative training on trigrams with the help of an example. From various literature surveys it was noted that a linear model representation for training the language models was an ideal place to start. The linear models require estimation of the parameter vector which was used as weights for the input feature vectors. This parameter vector was computed using a variation of the perceptron algorithm. It was found that the performance of the ASR system improved by 1.3% absolute by applying discriminative training using the perceptron algorithm [2].

5. REFERENCES

[1] Kai-Fu Lee, Mingjing Li, Zheng Chen, "Discriminative Training on Language Model", *Microsoft Research* October 2000.

[2] B.Roark, M.Saraclar, M.Collins, "Corrective Language Modeling For Large Vocabulary ASR with the Perceptron Algotithm" *Proceedings Acoustics, Speech, and Signal Processing, 2004*

[3] M.Collins, "Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms" In *Proceedings of the Conference on Emperical Methods in Natural Language Processing*, pages 1-8,2002

[4] J. Picone, "Fundamentals Of Speech Recognition," http://www.cavs.msstate.edu/hse/ies/publications/courses/ece_8463/

[5] X. Huang, A. Acero, and H.W. Hon, *Spoken Language Processing - A Guide to Theory, Algorithm, and System Development*, Prentice Hall, ISBN: 0-13-022616-5, 2001

[6] J. Deller, et. al., *Discrete-Time Processing of Speech Signals*, MacMillan Publishing Co., ISBN: 0-7803-5386-2, 2000.

[7] Daniel Jurasfky, James H. Martin, Speech and Language Processing, Prentice Hall, 2000.

[8] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification, Second Edition*, Wiley Interscience, ISBN: 0-471-05669-3, 2000.

[9] Saraclar, M.; Roark, B., "Joint Discriminative Language Modeling and Utterance Classification", In *Proceedings Acoustics, Speech, and Signal Processing, 2005*.