Go Beyond Trigram Language Modeling with Support Vector Machines

Jerry Xiaojin Zhu
School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213
zhuxj@cs.cmu.edu

February 5, 2001

Abstract

We propose research on a novel method to improve language modeling for automatic speech recognition. The problem of language modeling is to assign probabilities to any sentence, such that well-formed, meaningful sentences get high probabilities, while ungrammatic, nonsense ones get low probabilities. Traditional trigram language models are unable to model sentence level syntactic and semantic correlations. They often assign high probabilities to ill-formed sentences, which leads to recognition error. Our goal is to automatically discover these sentence level correlations, and use them to improve language modeling to reduce recognition error. We will first train a support vector machine classifier to learn a decision boundary between well-formed and ill-formed sentences, where the feature space is suitably chosen to express sentence level correlations. With the trained support vector machine, we determine whether a new sentence should be classified as ill-formed, and how confident we are about the classification. If the sentence is classified as ill-formed, we then decrease its language model probability by a function of our confidence. The function is chosen in a discriminative training fashion to minimize speech recognition error. We anticipate our method to improve automatic speech recognition accuracy, which is very important for the wide acceptance of speech recognition technology. The results of this research will be disseminated through publications.

1 Introduction

Automatic speech recognition (ASR) systems offer an attractive interface between human and machines. State of the art ASR systems have begun to show practical impact in situations where the user is physically challenged to use a keyboard or see a display; in hands-free eye-free applications such as driving assistant; or in telephone based automatic information services where keyboard input is limited. However these applications are usually very limited in acceptable vocabulary, syntax, or even speaking style. We still cannot talk to a computer as fluent as HAL-9000 in "2001: A Space Odyssey". Computers simply cannot recognize large vocabulary spontaneous speech, e.g. conversations in our everyday life, with enough accuracy, let alone the subsequent much harder natural language understanding, reasoning, and responding problems.

One reason for this lack of accuracy can be ascribed to the deficiency of a major module within ASR systems called the language model. Intuitively, a language model contains all the knowledge the system knows about language usage. Ideally the language model should assign high probability scores to all well-formed sentences (fluent, meaningful ones), and low probability scores to ill-formed sentences (ungrammatic, nonsense ones). An ASR system uses these probability scores to weigh acoustically similar candidate sentences in order to find the best decoding for a human utterance. For example, two candidate sentences for the utterance "It's hard to recognize speech." might be the correct sentence itself, and "It's hard to wreck a nice beach.". A good language model should give a high probability score to the first candidate sentence, and a much lower probability score to the second one. But to pack the complete language usage knowledge in a computational model is very hard. ASR researchers have to enormously simplify the language model to make it practically viable. Because of this simplification, language models are more lax than we would like it to be. It still gives high probability scores to well-formed sentences, but it might give higher scores to some ill-formed sentences, which leads to recognition errors. Creating better language models is therefore an important problem for ASR. The goal of this research is to improve language modeling to reduce ASR recognition errors.

We don't create a better language model from scratch. Rather, we start from an existing state-of-the-art language model and improve upon it. Specifically, we will work on an algorithm that discriminates between well-formed sentences and ill-formed sentences. With this algorithm we penalize the scores of such ill-formed sentences, and hence improve recognition accuracy. The outline of our approach can be divided into two phases:

- In phase 1 we solve a classic binary classification problem. First we view well-formed and ill-formed sentences as two distinct classes. We collect samples from each class. Then we train a support vector machine (SVM) to find a good decision boundary between the two classes. With the SVM we can compute the classification and margin (confidence) for any unseen new sentences.

- In phrase 2 we classify ASR candidate sentences with the SVM. Those sentences classified as ill-formed with large margins will have their language model probability scores penalized. The amount of penalization is tuned to minimize empirical ASR recognition error.

The result of this research, assuming successful, will be a new algorithm that improves upon existing language models to give ASR systems better recognition accuracy. The algorithm will have the following desirable properties: it is easy to implement with little coding; it runs fast during decoding time, which is crucial to ASR systems; it acts as a post processing and is self contained, can be plugged in or taken out of an ASR system without extensive changes to the existing language model or other parts. The result will be delivered in the form of conference/journal publications to other ASR researchers.

The research proposed here will help building better ASR systems, which have huge potential impact on our society. It will improve the quality of life for people with disability by providing voice control of home appliances; It will enable more natural and effective conversations between human and information agents over the phone; Eventually we will be able to talk to computers as we do to human audience today.

2 Related Work

Over the last decade, there has been extensive work on language modeling (see e.g., [1], [2], [3], [4], [5], [6]). A detailed survey of recent literature on language modeling can be found in [7]. Our approach differs from other work in the field in three important technical aspects.

1. We consider every sentence as an integrated unit. Most language models operate on subsentence levels like trigrams (word sequences of length 3) for pragmatic reasons. By doing so they lost the ability to model phenomena that range over a whole sentence, such as some syntactic and semantic constraints. Being able to model sentence level phenomena is essential to a better language model.

2. We find the deficiency of the existing language model with a machine learning approach, which is less ad hoc than relying on human judgement.

3. We penalize ill-formed sentences in a discriminative training fashion. That is, we train to minimize recognition error rate instead of maximizing the likelihood of the language model. Discriminative training has been shown to perform better in practice.

Recently Rosenfeld [8] introduced a whole sentence maximum entropy language model which is capable of model sentence level phenomena. Following the work, Chen and Rosenfeld [9] suggested a method to systematically explore the difference of any computable aspects (called features) between an existing language model and the empirical distribution of training sentences, and exploits the differences to improve the language model. These features can include sentence level features such as sentence length, syntactic structure of the sentence, etc. Zhu and Rosenfeld [10] applied the method to a set of sentence level linguistic features and showed some improvement in language model likelihood. However their work relied on human judgement to specify a family of computable features, on which the differences are sought. Human judgement, although usually judicious, is often highly focused on a small area of the problem space. Our proposed approach instead would use a machine learning method to automatically find a (possibly) complex feature that exhibits the difference most. We believe this will be a good complement to the previous work. Furthermore, [9] [10] optimized language model likelihood, while we try to minimize recognition error directly.

Cai et al. [11] also started from a set of manually picked features which are believed to capture the semantic coherence of a sentence. But instead of using the features as they are for language modeling, they fit a generalized additive model with smoothing splines. This is essentially a step toward letting the data decide how to best use the features. But their initial set of only six features are too limited to allow enough freedom in generalized additive model fitting, while we are going to start with a much broader choices. They also optimized language model likelihood.

Stolcke et al. [12] trained an 'anti' language model, not from well-formed sentences but from decoded candidate sentences. Since most candidate sentences contain errors originated from the language model, the anti-language model is specialized in finding the deficiency of the original language model. This is an example of applying machine learning approaches. However unlike our approach, their model cannot capture sentence level phenomena. They also optimized language model likelihood.

Perhaps the most similar work is by Eneva et al. [13]. Their work corresponds to phase 1 of our proposal, where they trained a classifier between well-formed and ill-formed sentences using decision trees and neural networks. But they didn't carry on the work to phase two, i.e. using the classification to improve speech recognition. We also plan to use a different classifier, namely SVM which has been proven to work well in text categorization tasks.

Most work in language modeling has the goal of maximizing model likelihood, which is Bayes optimal in theory. But in practice none of the components in an ASR system is perfect, which violates the Bayes optimal assumption. So discriminative training on language model, which directly minimizes training error, may be desirable. There hasn't been much work in this direction yet. Chen et al. [14] modified the language model probabilities of short word sequences if the sequences are misrecognized by ASR. This modification is limited to a lower level. In addition, other components often make recognition errors too and the language model shouldn't be penalized for such errors. Paciorek et al. [15] converted the discriminative language model training problem into an error rate optimization problem in feature weight space, and use a form of hill climbing to find a local minimum. The features are manually picked and there are severe overfitting and local minima problems with a modest number of features. Our approach will be more robust since we have very few parameters to tune, as we will see below.

3 Approach

In ASR systems, the final processing step is 'N-best list rescoring'. Before this step, the recognizer has generated N candidate sentences which are acoustically similar to the input utterance (but may not make sense syntactically or semantically). At this step the recognizer's task is to pick the best sentence both in acoustics and language. The N candidate sentences are called an N-best list, see Figure 1. Formally, each candidate sentence s has an acoustic score $P(u|s)$ which indicates its acoustic proximity to the input utterance u. N-best list rescoring is to find the best sentence s_u^* that maximize the *a posteriori* probability

$$s_u^* = arg\max_s P(s|u) = arg\max_s P(u|s)P(s) \quad \text{, by Bayes' rule}$$

Therefore we need a *language model* score $P(s)$ to multiply to the acoustic score and re-rank the N-best list to get the best sentence. $P(s)$ ideally should assign high scores to all well-formed sentences and low scores to all ill-formed sentences. This is very hard, partly because we don't yet know how to put human intuition of good and bad sentences in a computational model. In practice, the so called *trigram language model* is widely used because of its simplicity. In trigram language models, the probability of a sentence is the product of the conditional

$\log P(u\|s)$	candidate sentence s
-5522539	it *sites* class *eyes* quality of life as problems
-5556088	it *sites* class size quality of life *has* problems
-5556088	it cites class size quality of life *has* problems
-5622228	it *sites klas eyes* quality of life *has* problems
-5653812	it *sites* class size quality of life as problems
-5653812	it cites class size quality of life as problems
....	*(many, many other candidate sentences)*

Figure 1: The N-best list for utterance "It cites class size quality of life as problems." Errors are highlighted

probabilities of its words, conditioned on previous two words respectively:

$$P(s) = P(w_1, \ldots, w_n) = \prod_{i=1}^{n} P(w_i|w_{i-2}, w_{i-1})$$

and each conditional probability is estimated by counting a large text collection, or corpus:

$$P(w_i|w_{i-2}, w_{i-1}) \approx \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})} \tag{1}$$

Albeit popular, trigram language models have severe deficiencies, notably its inability to model long range, sentence level phenomena. This problem roots in their short range (two previous words) conditional independence assumption. Intuitively if we chop a sentence into several large chunks and scramble the order of the chunks, the resultant sentence will most likely have comparable trigram language model probability as the original sentence, but will be nonsense to a human reader. In our prior work, we have tried to incorporate sentence level linguistic and semantic information into language models and showed some improvement [10] [11]. In this work, we will extend our research with machine learning methods to automatically improve upon trigram language models [1].

3.1 Technical Approach

Our approach takes a whole sentence as a unit. It first investigates the difference between well-formed and ill-formed sentences, then use the difference to improve language modeling for ASR.

3.1.1 Phase 1: investigating the difference

The probability distribution of well-formed sentences is unknown. But we can easily acquire samples (here and in the sequel, a sample is a complete sentence) from that distribution. Indeed, every sentence published in books, newspapers, on the web, or spoken on radio and TV broadcast, is a sample from the distribution. In the following we call well-formed sentences *true sentences* to indicate their origin.

[1] In theory our method can be used to improve any kind of language model.

how many different people once if you're going to interrupt you right
i just want approval ratings are drawn to the rescue operations of the films that were going too fast
and we hope to move because of nafta culture has reached out to walk down the authority to stop the deity republican conventions with a star power
paranoia about each other for each of you
yes hello
believers say who needs a lawyer from a unk very impressed with such and such is not a time magazine that we just don't believe
he took over or is it by all human beings
danny rolling
and we can put on trial
expect the fiction within the white house and the drag racing school steered a costly and time is up to eight bucks or part of the abortion plank

Figure 2: Sample sentences generated from a trigram language model

For ill-formed sentences, we focus on a subset of them, namely those generated by a trigram language model with typical probabilities, since they are in the N-best lists. In the following we call them *trigram-generated sentences*. It is easy to use the trigram language model as a generative model and generate sentences as follows: select the first word randomly according to word frequency; select next word randomly according to the trigram probability (1) and so on. Figure 2 shows a few sentences generated in this fashion. They (especially the longer ones) are very distinct to a human reader.

We pose the problem of automatically finding the difference between true and trigram-generated sentences as a binary classification problem. (Note in fact the two classes completely overlap since both is capable of generating any sentence. But their probability distributions are very different.) The samples mentioned above serves as positive and negative examples.

There are many methods for binary classification, and we choose support vector machines (SVM). A SVM finds a linear decision boundary that separates the positive and negative examples in an augmented feature space, such that the distance (the so called margin) from the decision boundary to the closest positive or negative example is kept as large as possible. This maximization of margin gives SVM good generalization performance. In addition, SVM has a computationally efficient way to augment the feature space via so called kernel functions. For details see e.g. [16] [17]. We choose SVM because

- It is the state-of-the-art classification method in text categorization tasks [18], which is closely related to our problem;
- Its kernel function gives us potential ways to express word interactions on a sentence level;
- It produces a single number (the margin) when classifying a new sentence. The number can easily be converted to a posterior probability or used in the subsequent discriminative training.

The performance of SVM is influenced by our design of two things: the feature representation of the sentences, and the kernel function. Traditionally people have used 'bag-of-words' feature presentation [19]. But it might not give us the best performance since this representation ignores word order information completely, which intuitively is important in our domain.

We plan to investigate a variety of feature representations, including bag-of-words, adjacent word pairs, distant word pairs, word co-occurrence statistics, etc. To augment the feature space, we will investigate the polynomial kernel and possibly some other kernels.

Once we have trained a SVM, we can classify a new sentence as either a true sentence or a trigram-generated one. We also get the raw (unthresholded) output $f(s)$ of the SVM

$$f(s) = \sum_i y_i \alpha_i k(s_i, s) + b$$

which is related to the confidence or the margin. We will use $f(s)$ to improve N-best list rescoring.

3.1.2 Phase 2: improving speech recognition

The intuition that motivated this research is that ASR recognition errors are usually caused by a slack language model which assigns too high a probability to an ill-formed sentence. Therefore a natural strategy would be as follows: if a sentence looks more like trigram-generated then true, we would decrease the language model probability. The judgement is exactly the binary classification problem we have seen; the amount to decrease the probability will be a function of our confidence, or in SVM's case, the raw output $f(s)$. $f(s)$ is a real number, the larger the magnitude the more confident we are. Platt [20] suggested a method to convert $f(s)$ to a posterior probability $P(\text{trigram-generated}|s)$. Or we may use an arbitrary function of it, $g(f(s))$ to create a modified language model probability $P'(s)$ with an exponential function:

$$P'(s) = \frac{1}{Z} P(s) e^{\lambda * g(f(s))}$$

where Z is a normalization factor and λ is the parameter.

With this modified language model, the best sentence s_u^* given an utterance u is:

$$\begin{aligned} s_u^* &= arg \max_s P(u|s) P'(s) \\ &= arg \max_s \log P(u|s) + \log P(s) + \lambda * g(f(s)) \end{aligned}$$

We are interested in discriminative training, i.e. tuning λ to directly minimize the error on the training set. Let $e(s)$ be the number of errors in decoded sentence s. The total error on the training set is

$$E(\lambda) = \sum_{u \in \text{training}} e(s_u^*)$$

E is a complex function of λ. We can find an optimal λ^* corresponding to a local minimal E by simple hill climbing, or simulated annealing. Note here we use different training data than in phase 1: now we use a set of N-best lists (candidate decoding sentences with acoustic scores) as the training data. Once we find the optimal λ^*, we perform N-best rescoring on a new utterance and decode the utterance as

$$arg \max_s \log P(u|s) + \log P(s) + \lambda^* * g(f(s))$$

which we expect to have less error than the decoding with the original trigram language model:

$$arg \max_s \log P(u|s) + \log P(s)$$

3.2 Anticipated Results

- In phase 1, we expect to see the SVM, with appropriately selected feature representation and kernel, achieve good classification result on the true vs. trigram-generated sentences. The trained SVM is the product of this phase.
- In phase 2, we expect reduced recognition error with the discriminatively trained N-best list rescoring process. The SVM in phase 1 and the optimal parameter λ^*, together with the process described above, are the final result of the proposed research.

3.3 Strategy of Evaluating

- In phase 1, we will measure the SVM classification error rate on a hold-out set of true and trigram-generated sentences. The hold-out set can be a randomly selected 30% of the total samples, and the rest 70% is the training data for the SVM. We can repeat the process several time. The average classification error rate should be statistically significantly better than random.
- In phase 2, we measure the error reduction on a hold-out n-best lists, before and after modifying the trigram language model with the SVM margin. Similarly, the hold-out set is a random split of all available N-best lists. The process is repeated several times. The average error reduction should be significantly larger than zero.

4 Work Plan

The work is divided into 4 stages:

1. Feb. 5-9 (1 week) Prepare the true sentence and trigram generated sentence data sets; prepare the N-best list data set; Set up SVM training program.
2. Feb. 12-march 9 (4 weeks) Work on phase 1, train and tune the SVM. This involves finding the best feature representation and kernel function.
3. March 12-23 (2 weeks) Work on phase 2, discriminative training and test on N-best lists.
4. March 26-30 (1 week) Project write up.

5 Summary

We proposed a novel method to improve language model for ASR with a machine learning approach. The proposed research is part of a larger research effort carried out by the statistical language modeling group at Carnegie Mellon University. Our long term goal is to develop better language models by incorporating long range, sentence level phenomena into the model, and by combining human linguistic knowledge with data driven modeling approaches.

The social impact of perfected ASR systems will be enormous. Voice web surfing, car navigation, telephone information systems, and robot control are just a few examples of what this new human-computer interaction modality begins to bring. As an analogy, consider what graphic displays and mice have brought to the flourish of personal computers today. We have reasons to believe ASR will have similar impact to future computing. Yet ASR needs support from good language models. Our proposed research is one step closer towards this end.

References

[1] Slava M. Katz. Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 35(3):400–401, March 1987.

[2] Frederick Jelinek and Robert L. Mercer. Interpolated estimation of Markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*, pages 381–397, Amsterdam, The Netherlands: North-Holland, May 1980.

[3] Ian H. Witten and Timothy C. Bell. The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. *IEEE Transactions on Information Theory*, 37(4):1085–1094, July 1991.

[4] Reinhard Kneser and Hermann Ney. Improved backing-off for m-gram language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, volume I, pages 181–184, Detroit, Michigan, May 1995.

[5] Frederick Jelinek. *Statistical Methods for Speech Recognition.* MIT Press, Cambridge, Massachusetts, 1997.

[6] Ronald Rosenfeld. A maximum entropy approach to adaptive statistical language modeling. *Computer Speech and Language*, 10:187–228, 1996. longer version published as "Adaptive Statistical Language Modeling: A Maximum Entropy Approach," Ph.D. thesis, Computer Science Department, Carnegie Mellon University, TR CMU-CS-94-138, April 1994.

[7] Ronald Rosenfeld. Two decades of statistical language modeling: Where do we go from here? *Proceedings of the IEEE*, 88(8), 2000.

[8] Ronald Rosenfeld. A whole sentence maximum entropy language model. In *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding*, 1997.

[9] Stanley F. Chen and Ronald Rosenfeld. Efficient sampling and feature selection in whole sentence maximum entropy language models. In *ICASSP-99*, Phoenix, Arizona, 1999.

[10] Xiaojin Zhu, Stanley F. Chen, and Ronald Rosenfeld. Linguistic features for whole sentence maximum entropy language models. In *Proceedings of the European Conference on Speech Communication and Technology (Eurospeech)*, Budapest, Hungary, 1999.

[11] Can Cai, Larry Wasserman, and Roni Rosenfeld. Exponential language models, logistic regression, and semantic coherence. In *Proceedings of the NIST/DARPA Speech Transcription Workshop*, May 2000.

[12] A. Stolcke, H. Bratt, J. Butzberger, H. Franco, V. R. Rao Gadde, M. Plauche, C. Richey, E. Shriberg, K. Sonmez, F. Weng, and J. Zheng. The SRI march 2000 Hub-5 conversational speech transcription system. In *Proceedings of the NIST Speech Transcription Workshop*, College Park, MD, 2000.

[13] Elena Eneva, Rose Hoberman, and Lucian Vlad Lita. Semantic coherence, 2000. Course Project Report for 15781 Machine Learning Fall 2000, Carnegie Mellon University.

[14] Zheng Chen, Kai-Fu Lee, and Ming-Jing Li. Discriminative training on language model. In *Proceedings of the ICSLP*, 2000.

[15] Chris Paciorek and Roni Rosenfeld. Minimum classification error training in exponential language models. In *Proceedings of the NIST/DARPA Speech Transcription Workshop*, May 2000.

[16] C. Burges. A tutorial on support vector machines for pattern recognition. *Data Mining and Knowledge Discovery*, 2(2):955–974, 1998.

[17] N. Cristianini and J. Shawe-Taylor. *An introduction to support vector machines and other kernel-based learning methods.* Cambridge University Press, 2000.

[18] T. Joachims. Text categorization with support vector machines: Learning with many relevant features. In *Proceedings of the European Conference on Machine Learning*, 1998.

[19] Andrew McCallum and Kamal Nigam. A comparison of event models for naive bayes text classification. In *AAAI-98 Workshop on Learning for Text Categorization*, 1998.

[20] J. Platt. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In A. Smola, P. Bartlett, B. Scholkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers.* MIT Press, 1999.