Using Evolutionary Optimization to Improve Markov-based Classification with Limited Training Data

Timothy Meekhof
University of Idaho
Moscow, Idaho, USA
timothym@cs.uidaho.edu

Robert B. Heckendorn
University of Idaho
Moscow, Idaho, USA
heckendo@uidaho.edu

ABSTRACT

Bayesian classification using Markov model analysis of token strings is used in many areas such as computational linguistics, speech recognition, and bioinformatics. Unfortunately, for many problems, the available data sets are too small to accurately estimate the large number of parameters in a Markov model. In our work, we explore the possibility of using string space transformations to reduce the perplexity of the modeling problem and thereby improve model performance. The set of all possible string-to-string transformation functions is very large. By using a genetic algorithm to search for transformation functions that improve the performance of a Markov-based classifier, we are able to construct a classifier system that performs better than the Markov classifier alone. We go on to demonstrate the improved performance on the problem of classifying English and Spanish character strings, where training set size is arbitrarily limited.

Categories and Subject Descriptors

I.5 [**Pattern Recognition**]: Design Methodology

General Terms

Algorithms, Experimentation

1. TRANSFORM, THEN CLASSIFY

Markov-chain classification of strings of symbols has important practical significance for such diverse applications as natural language processing [2], protein analysis [4], and speech recognition [1]. In many applications, though, size of the available data for training a string classifier is small, resulting in poor performance, because there are not enough training samples to reliably estimate all of the model parameters.

To overcome the problem of insufficient data, we make use the concept of string transformation, in which we substitute some of the substrings within our training set with single tokens, often mapping several substrings to the same token.

If some substrings are eliminated from the input data and some classes of tokens are replaced by a single representative token, the average perplexity of the data is reduced.

Perplexity is a measure of the difficulty of predicting the next token in a string, and is based on the entropy of the training set, $\mathcal{T}$. The order n perplexity of the set $\mathcal{T}$ is defined as [1]:

$$\mathrm{PP}_n(\mathcal{T}) = 2^{H_n(\mathcal{T})} \tag{1}$$

where $H_n(\mathcal{T})$ is the entropy of the training data using an order n Markov model. Perplexity measures the number of different possible tokens that can expect to occur next in sequence in a Markov chain, and directly affects the difficulty of constructing the Markov model.

The number of parameters in a Markov chain model is approximately γ^m, where γ is the average perplexity of the training set and m is the order of the model. Obviously, reducing γ can have a large impact on the amount of data necessary to train a Markov model.

While reducing the perplexity of the data makes construction of an effective Markov model more tractable, it may also eliminate the very information that is necessary for performing the task of the classifier. The task is to find a string transformation function that reduces perplexity and also improves the classifier's performance as measured by percentage correct classifications on held out data, if such a function exists. Because the space of possible string transformations is very large, we make use of a genetic algorithm (GA) to search for string transformations that improve the performance of our classifier.

2. STRING TRANSFORMATIONS

To reduce the perplexity of our training data, we use a string transformation function, a mapping $\pi : \mathcal{S} \to \mathcal{S}$. In our case, the string transformations are constructed as a set of simple mapping operations such as the following:

$\tau \to$ the	*the* becomes τ.
$a \to e$	any *e* is replaced by *a*.
$a \to i$	any *i* is replaced by *a*.
$q \to qu$	any *qu* is replaced by *q*.
$\omega \to$ t%r	*t* followed by any character followed by *r* becomes ω.

A string transformation function is a list of one-to-one and many-to-one mappings like the ones listed above. To transforma an input string, the program scans the string. At each token of the input it scans through the transformation list; the first pattern that matches the token(s) will output its token and the current position in the input string will advance

$\vert T_m \vert$	Base VLMC	Random	Best
25 characters	52.0%	60.5%	84.4%
50 characters	71.6%	67.9%	88.6%
100 characters	84.8%	79.0%	91.7%

Table 1: Classifier Performance on Held Out Data (Average over 3 tests)

by the length of the pattern. If no pattern matches the current position, then the current input token is output and the scan position advanced by one. If the rules given above form a definition of the transformation function π, then the following statements would be true:

π(theory)	=	τory
π(happy)	=	happy
π(torture)	=	$\omega\omega$a
π(thedogisquiet)	=	τadagasqaat

The transformation need not be reversible and does not preserve all the information provided in the original strings. There is also no guarantee that any particular transformation function will result in an improved Markov classifier, even if it reduces the perplexity of the training set. An extreme example is a that maps all inputs tothe same token. The result has a perplexity of 0, but makes a poor classifier.

The task is to find string transformations that reduce the perplexity of the training set and improve classification.

The set of possible transformation functions is unbounded and suggests itself to the application of a genetic algorithm search, as discussed at length in [3].

3. EXPERIMENT

To examine the behavior of our method, we collected samples of English and Spanish text. The task given our system was to classify input strings as either English or Spanish. Though there is certainly enough extent data This problem would seem to be quite easy.

Our purpose in this project is to determine how *small* a random sample of text can be used to achieve a given level of performance on the classification task. Our reference case for our experiments is a Variable Length Markov Chain (VLMC) model.

Genetic search through the space of string transformation was used. Each member of the population is a set of transformation operations. A steady state algorithm and a population of 250 elements was used.

The numbers in Table 1 are the percentage of correct classifications produced by the classifier against the held out data set, $\mathcal{T}_H$. The *Base VLMC* column gives the performance of the VLMC classifier without any string transformation operations; the column labeled *Random* gives the classifier's performance on the first, random population of the GA; and finally the *Best* column reports the best of the final population of the GA.

The final performance of the classifiers is better than the base performance, and demonstrates that the GA has succeeded in finding a transformations that decrease the perplexity of the data without removing the information that improves classification.

4. CONCLUSION

In this paper we have described a technique to improve the performance of a string classification system using limited training data. To reduce data perplexity over a limited training set, we introduce the concept of generalized string transformation functions. By reducing the perplexity of the data, the string transformations reduce the average number of necessary parameters in the Markov model and increase the reliability of the remaining parameters.

However, there is no guarantee that transforming the data will result in a better classifier, even if the resulting data has lower perplexity. A transformation must be found that improves classifier performance. To accomplish that, we use a genetic algorithm to search the space of string transformations, in order to find a set of string transformations that do in fact improve classifier performance.

The results of the experiment clearly indicate that the genetic search for string transformations was able to improve classification accuracy over the base VLMC model.

$\vert T_m \vert$	Base VLMC	Random	Best
25 characters	21.2	15.4	17.7
50 characters	21.4	6.7	13.7
100 characters	21.5	5.3	11.4

Table 2: Perplexity of the English Data

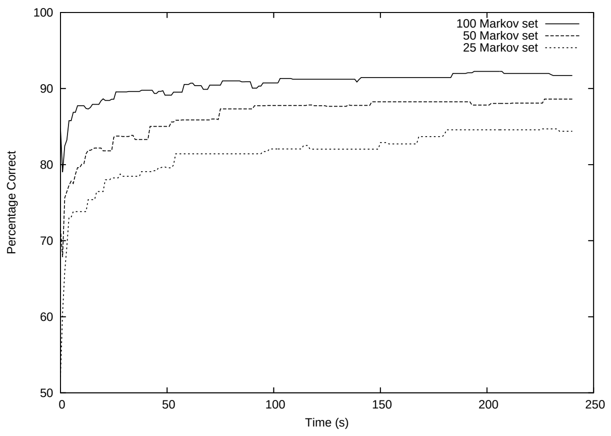

Figure 1: Percentage of correct classifications in the held out dataset over time

Acknowledgments

This resarch was supported by NIH Grant #P20 RR16448 from the COBRE Program of the National Center for Research Resources.

5. REFERENCES

[1] L. R. Bahl, F. Jelinek, and R. L. Mercer. Likelihood approach to continuous speech recognition. *Transactions on Pattern Analysis and Machine Intelligence*, 2:179–190, 1983.

[2] P. F. Brown, V. j. Della Pietra, P. V. deSouza, J. C. Lai, and R. L. Mercer. Class-based *n*-gram models of natural language. *Computational Linguistics*, 18:467–479, 1992.

[3] A. Eiben and J. Smith. *Introduction to Evolutionary Computing*. Springer-Verlag, Viena, Austria, 2003.

[4] A. K. G. M. R. Durbin, S. Eddy. *Biological Sequence Analysis*. Cambridge University Press, Cambridge, United Kingdom, 1998.