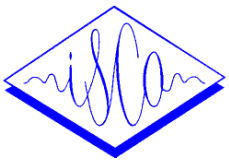

4th European Conference on
Speech Communication and Technology
EUROSPEECH '95
Madrid, Spain, September 18-21, 1995

AUTOMATIC CLUSTERING OF WORDS FOR PROBABILISTIC LANGUAGE MODELS

Loreta Moisa[1] *Egidio Giachin*[2]

[1]Politechnic University of Bucharest
Str. Splaiul Independentei, nr 313 - Bucuresti, Romania
[2]CSELT - Centro Studi e Laboratori Telecomunicazioni
Via Reiss Romoli, 274 - 10148 Torino, Italy

Abstract

In this work we compare different methods for clustering words into equivalence classes within a bigram language model, for a specific-domain recognition task (train timetable enquiry). Though the perplexity values obtained by the various methods differ, the word error rates eventually achieved are very similar. We examine this behavior in the light of the word usage peculiarities present in these types of tasks.

10.21437/Eurospeech.1995-311

1 INTRODUCTION

The development of a probabilistic language model for speech recognition requires a training database, from which the system lexicon is also derived. For some large-lexicon tasks the number of parameters to train may be so high that there are not enough training data to obtain a sufficiently reliable model. To reduce the number of parameters to be estimate (thereby increasing their statistical robustness) words may be clustered into classes and class-based rather than word-based models are trained. The critical issue is then the choice of a satisfactory word clustering method. Several methods have been proposed, based on information-theoretic criteria [3], on simulated annealing [5], and other principles. Recently an optimal procedure, in the sense that it minimizes perplexity, has been described [9]. Though this is theoretically the best choice, it is known that perplexity alone is not the only factor that affects a recognizer's performance, thus one might argue that different statistically robust methods may turn out to provide equal or better results in practice.

In this study we implemented algorithms based on the above methods and some variants, and compared their performance on an 800-word continuous speech recognition task, referring to train timetable database enquiry [4]. Performance is evaluated using a bigram language model; both perplexity and word error rate are considered. All methods provide improvements over manual clustering. Experimental results indicate that the performance improvements depend on the choice of important parameters (number of classes, etc.). However, when pushed to their best, all methods provide comparable recognition performance. The algorithms actually share some common behavior, which is related to the peculiar statistical distribution of words found in spontaneous speech sentences referring to specific-domain tasks.

2 CLUSTERING METHODS

In its more general acception, the use of word classes may be viewed as a double stochastic process (emission of a class given the preceding ones, and emission of a word given the class) and thus requires a HMM treatment similar to what is done for acoustic recognition. In this work it is assumed that each word can belong to only one class, which makes word clustering a partition. This is sufficient for the task under study and permits to adopt simpler algorithms. We have experimented with the following ones: the *Maximum likelihood* method, both in its baseline and *Leaving one out* versions, the *Merge* method, and a method based on *Simulated Annealing*. When necessary, smoothing has been adopted to furtherly enhance the statistical robustness of language models. In all cases, linear smoothing has been used [10].

Clustering is "automatic" but for a small number of classes. A specific-domain task, in fact, includes groups of words that obviously play exactly the same role in a sentence's syntactic structure (e.g., in our domain, the names of cities, stations, days of week, etc.). If one assumes that the word usage within each group has no preference (that is, "Torino" has the same probability as "Milano"), the identification of these groups is best done manually. Seven of these "special" classes have then been identified; words of any special class cannot be moved to other classes, nor can words of other classes be moved in the special ones.

2.1 Maximum likelihood

The Maximum likelihood method is derived from [9]. This procedure aims at maximizing the likelihood of the training database (hence at decreasing its perplexity), for a given number M of classes into which the words have to be partitioned. Starting from an initial partition, the procedure iteratively moves each word from a class to other classes, finding the move that corresponds to the maximum increase of the training database likelihood. The procedure stops when no improvement is obtained. The optimum number of classes must be found by experiment. Several issues have been examined:

- The partition output by the algorithm depends on the initial one. Several choices are possible (manual classification, all words in one class, all words in one class except for the K ($K < M$) most frequent words, etc.).
- The best move at each cycle may be taken for each examined word, or the absolute maximum may be taken (by examining at each cycle all words and all moves)

Speed of convergence is a critical issue (as it is for the algorithms below); implementation has to be carefully designed in order to reduce computational requirements.

In practice, what is optimized is not likelihood but a quantity Q derived from it by expressing bigram probabilities through occurrence counts, dropping terms independent of clustering, and suitably regrouping the remaining terms. This leads to [9]

$$Q = -\sum_{c_1,c_2} N(c_1,c_2)\log N(c_1,c_2) + 2\sum_{c} N(c)\log N(c)$$

This quantity varies from 0 (good) to ∞ (bad), so it must be minimized. The c's vary over the whole cluster set and $N(.)$ represents the number of occurrences of the argument in the database. (It is assumed that $0 \log 0 = 0$.)

2.2 Leaving-one-out

This procedure also derives from [9]. The Maximum likelihood method uses the same database to determine the clustering and to estimate probabilities. Though this works properly in practice, to avoid biases on the side of the training database a different validation database should be used for the second goal. The leaving-one-out technique permits to exploit the same database both as a training and validation set. The above algorithm can therefore be modified accordingly, which leads to a different expression for the quantity to be optimized [9] (it includes the effect of smoothing).

Since empty classes do not constitute a problem, this method can be designed so as to consider the final number of classes as another parameter to be optimized and automatically determined. This property was not used in the experiments reported here.

2.3 Merge

This algorithm [3] is also based on maximizing the likelihood of the training set (which is shown to be equivalent to maximizing the mutual information between adjacent words), however it aims at this goal through a different way. Instead of moving words from one class to a different one, it starts with an initial partition in which each word is in a separate class, then it iteratively finds the two classes that, when merged, give rise to the greatest increase (or smallest loss) of mutual information between adjacent classes. It stops when the number of remaining classes is equal to a predefined number M.

Beside the basic version, some variants are possible. For example, the most frequent words may be prevented from merging to avoid "disturbances" by low frequency words.

2.4 Simulated annealing

The previous methods perform at each step the "best" move and tend therefore to reach a local minimum of perplexity. Simulated annealing, which derives from Monte-Carlo simulation of the thermodynamical process of heating and annealing of solids, permits some locally "bad" moves to take place, in the hope that the global maximum may eventually be found. This principle was used for word clustering in [5]. Let us briefly review it here.

The algorithm reproduces a Boltzmann machine. Given a configuration i, it iteratively generates a random configuration j in a neighborhood of i. A cost $P(i)$ is associated to each configuration. The goal is to minimize $P(i)$; a temporary increase of $P(i)$ may be accepted. If $\Delta P(i,j)$ represents the cost variation in the passage from i to j, then the probability $Pr(i \rightarrow j)$ that j is accepted is

$$Pr(i \rightarrow j) = \begin{cases} 1 & \text{if } \Delta P(i,j) \leq 0 \\ e^{-\Delta P(i,j)/T} & \text{if } \Delta P(i,j) > 0 \end{cases}$$

where T is a control parameter that simulates temperature. New configurations are tried until equilibrium is reached. T is decreased during the process, which stops when T is sufficiently small or $P(i,j)$ does not change significantly. The cost $P(i)$ is, in our case, the same quantity Q used in the Maximum likelihood procedure. The resulting algorithm is therefore as follows:

1. Start with an initial clustering C
2. Until the final temperature T_f is reached, do
   (a) Until equilibrium is reached, do
      i. Randomly choose a word w and a class c
      ii. Compute the cost variation $\Delta P(i,j)$ of moving w into c
      iii. If $\Delta P(i,j) \leq 0$ then accept the move
      iv. If $e^{-\Delta P(i,j)/T} > random(0,1)$ then accept the move
   (b) Change the temperature

This algorithm contains a double iteration: a significant number of configurations is tried for a fixed T, then T is decreased and the procedure is repeated. This success of this approach depends on a number of parameters that have to be experimentally determined:

- The initial temperature T_0. In the beginning, most changes must be accepted. T_0 is chosen so that the probability of acceptation is slightly less than 1.
- The final temperature T_f or the minimum rate of change of $P(i,j)$.
- The number of moves for a given T (i.e. for each inner cycle). We chose it so that in a cycle all words have been tried.

Clustering	Perplexity	Word error
NO CLUST	20.9	18.5%
MANUAL (192 classes)	42.7	21.2%
ML (250 classes)	25.8	17.7%

Table 1: Effect of clustering

- The rule for varying the temperature. To reach the global minimum T should decrease logarithmically, however this process is very slow. A linear decrease may be chosen in practice.

3 RESULTS

Results are evaluated on an 751 word spoken language task referring to railway timetable enquiry [4]. The training database [1] includes about 9,000 spontaneous speech sentences, amounting to about 60,000 words, collected from "naive" users through a PABX; the test set includes 1358 sentences. The recognizer [6] employs 310 context-dependent subword units modeled via 3 state discrete density HMMs.

The effect of clustering is shown in Table 1, where the test set perplexity and word error rate are reported for the no clustering case (actually one word per class, except the "special" classes of Sect. 2), a typical manual classification, and one of the best automatic clustering, obtained with the simple Maximum likelihood (ML) method. The automatic clustering provides a small improvement over the no clustering case; improvement is higher on more troublesome sentences. The manual clustering, though carefully worked out, actually performs even worse than no clustering. The relation between perplexity and word error is not always monotonic: the no class case has a lower perplexity than the ML case, but a higher word error.

3.1 Perplexity and word error

Fig. 1 shows the perplexity and word error for different numbers of final classes in the Maximum likelihood method. It can be seen that perplexity keeps decreasing, and will eventually reach the no-clustering value; however the error rate does not, and reaches a minimum for about 250 classes. This suggests that taking perplexity alone as a measure for a language model may lead to nonoptimal solutions, and that word error rate should be taken as the ultimate evaluation criterion. This is justified by the fact that the value of perplexity is affected by factors such as smoothing more than recognition accuracy.

3.2 Comparison of algorithms

A comparison between the different approaches can be seen from Table 2. There the perplexity and word error are reported for the Maximum likelihood (ML), Leaving-one-out (LOO), Merge (ME), and Simulated Annealing (SA) methods. All were pushed to their best performance by an appropriate choice of the final number of classes and recognition parameters (word

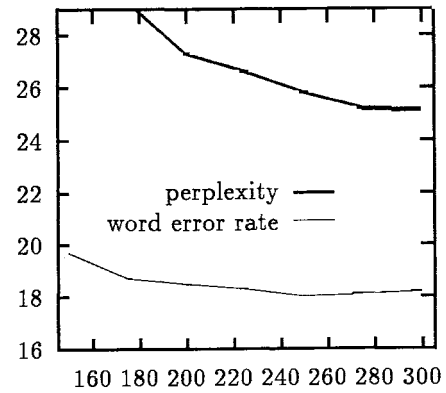

Figure 1: Perplexity and word error vs. no. of classes

Clustering	No. classes	Perpl.	W. err.
ML	250	25.8	17.7%
LOO	250	25.5	17.8%
ME	300	29.7	18.4%
SA	300	24.8	17.7%

Table 2: Comparison of different algorithms

penalty and model gain, i.e. the factor that multiplies the log prob given by the language model before adding it up to the acoustic log-prob). Though the perplexity values are rather different, the actual recognition accuracy achieved by the different clusterings (except maybe Merge) is very similar. Also, it was verified that the word error variation remains small for a large range of number of classes. Apparently, once a basic "good" clustering is found, the different methods produce only unimportant small variations.

As said in Sect. 2, some variants to the basic algorithms were implemented and tested. The ML algorithm was designed so as to classify words in an order according to their frequency of occurrence, and to neglect clustering of words with occurrence below a threshold. Also, it was modified so as to explore all pairs word–couple in the inner loop before choosing the best move, in the hope to gain higher perplexity reduction in the first steps (though at expense of a much longer execution time). The Merge algorithm was modified to privilege, in the first steps, the merging of low-frequency words in order not to disturb the most frequent ones. The Simulated Annealing algorithm was modified to keep into account the best move at each inner cycle. These variants did not produce any significant change in the results.

3.3 Clustering and word usage

In order to gain a deeper insight on how really the clustering works, we have investigated the classes that were produced out of the task lexicon. The word usage in our task is rather peculiar. Fig. 2 plots in a log-log scale the occurrence probability of words vs. their rank. For large databases it has been observed that these two quantities are roughly inversely pro-

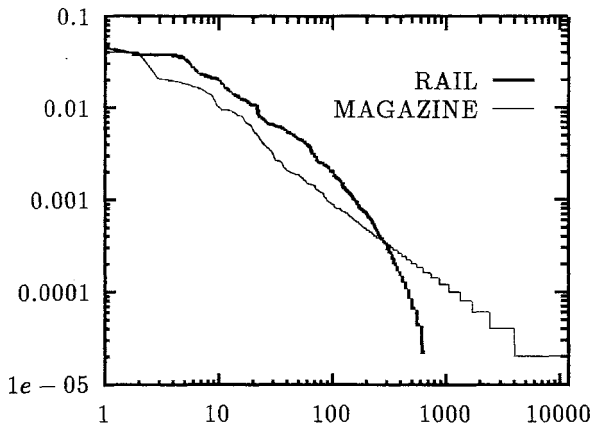

Figure 2: Word probability vs. rank

portional; the expected curve is then a straight line (a situation referred to as the Zipf's law). This, however, is not true for the spontaneous speech database referring to railway inquiries (thick line), while a database of comparable size drawn from an Italian magazine behaves as expected (thin line). The deviations of the railway database consist in a higher number of very frequently used words (less tilted upper left portion of the curve) and a higher number of sparsely used words that occur very rarely (more tilted lower right portion). These peculiarities do not seem to depend on the database size (the curve remains stable for a different range of sizes) and is probably typical of these kind of tasks. Under such conditions the various clustering algorithms behave according to a common pattern. They try to put the most frequent words each in a separate class (about 100–150 words are treated this way). Of the remaining words, some are clustered in a linguistically reasonable fashion, e.g. {*costa ammonta*} (both meaning "to cost"), {*andare recarmi*} ("to go"), {*verso non-dopo*} ("towards" and "not-later-than", used in temporal expressions), {*qualche alcune*} ("some"), {*viaggia viaggiano circola circolano*} ("it-, they-travel"), {*cena pranzo ristoro ristorazione*} ("lunch dinner restoration"), etc. Other clusters are clearly the result of two or more clusters that have been merged together due to the scarcity of available classes. Finally, most words that occur a low number of times are scattered at random and attached to words that apparently have nothing in common with them: the benefit of keeping them separate from one another wins over the confusion raised by their assignment to heterogeneous clusters. The clustering produced by the various algorithms differ mainly in this last action. Though it affects perplexity, it is not important for recognition; therefore the final accuracy performance is about the same.

The improvement over the no clustering and the manual clustering cases suggests that automatic clustering can be profitably used in specific domain spontaneous speech tasks like the one of the present study and, in that case, the choice of algorithms and parameters is not crucial. It should be noted that the improvement over the no clustering case is moderate, which argues in favor of using clustering in the initial development phase of a dialogue system, when the available training data are limited, and to switch to the no clustering case when a large enough training set has been collected. These results may not be extended to other tasks using a larger and more varied lexicon.

4 CONCLUSIONS

Four different algorithms for word clustering, together with some variants, have been compared on a spontaneous speech dialogue task referring to railway timetable enquiry. The performance are similar for the various algorithms, a fact that seems related to the peculiarities of word usage in this kind of task. They exhibit a sensible error rate improvement over the no clustering and the manual clustering cases, though perplexity would not always suggest that. As a next step, the behavior of these and other algorithms [2, 12] will be investigated in tasks with a larger and more varied lexicon and in the framework of a more complex language model (trigrams).

Acknowledgment

We wish to thank our colleague R. Gemello for providing support on the Simulated Annealing methodology.

References

[1] P. Baggia, E. Gerbino, E. Giachin, and C. Rullent, "Experiences of spontaneous speech interaction with a dialogue system", *CRIM/FORWISS Workshop*, München, September 1994.

[2] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, Wadsworth & Brooks, 1984.

[3] P. Brown, V. della Pietra, P. deSouza, J. Lai, and R. Mercer, "Class-based n-gram models of natural language", *Computational Linguistics*, Vol. 18, No. 4, 1992.

[4] D. Clementino and L. Fissore, "A man-machine dialogue system for speech access to train table information", *Eurospeech 93*, Berlin, September 1993.

[5] M. Jardino and G. Adda, "Automatic word classification using simulated annealing", *ICASSP 91*, Minneapolis, MN, May 1991.

[6] L. Fissore, E. Giachin, P. Laface, and P. Massafra, "Using grammars in forward and backward search", *Proc. Eurospeech 93*, Berlin, September 1993.

[7] F. Jelinek, "Self-organized language modeling for speech recognition", 1987, in K.-F. Lee, A. Waibel (Eds.), *Readings in Speech Recognition*, Morgan-Kaufmann, 1989.

[8] F. Jelinek, R. L. Mercer, and S. Roukos, "Principles of lexical language modeling for speech recognition", in *Advances in Speech Signal Processing*, S. Furui and M. M. Sondhi, Eds., Dekker, New York, 1992.

[9] R. Kneser and H. Ney, "Improved clustering techniques for class-based statistical language modelling", *Eurospeech 93*, Berlin, September 1993.

[10] H. Ney and U. Essen, "On smoothing techniques for bigram-based natural language modelling", *ICASSP 91*, Toronto, Ont., May 1991.

[11] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependencies in stochastic language modelling", *Computer Speech and Language*, Vol. 8, 1994.

[12] H. Späth, *Cluster analysis algorithms for data reduction and classification of objects*, Ellis Horwood, 1980.