A NOVEL TREE-BASED CLUSTERING ALGORITHM FOR STATISTICAL LANGUAGE MODELING

G. Damnati and J. Simonin
France Telecom CNET DIH/RCP,
2 av. Pierre Marzin, 22307 Lannion Cedex, France.
Tel.(33)2.96.05.13.88 / Fax.(33)2.96.05.35.30
e-mail: damnati@lannion.cnet.fr, simonin@lannion.cnet.fr

ABSTRACT

In this paper, a new method to cluster words into classes is proposed in order to define a statistical language model. The purpose of this algorithm is to decrease the computational cost of the clustering task while not degrading speech recognition performance. The algorithm provides a bottom-up hierarchical clustering using the reciprocal neighbours method. This technique consists in merging several pairs of classes within a single iteration. Experiments on a spontaneous speech corpus are presented. Results are given both in terms of perplexity and word recognition error rate. We obtain a large reduction in the number of iterations necessary to build a classification tree and thus a CPU time reduction in building the model as well as a reduction in both perplexity and word error rate.

1.INTRODUCTION

The purpose of statistical language modeling is to estimate the probability of sequences of words. According to the bigram approximation, the probability of such a sequence is the product of probabilities of each word of the sequence given its predecessor. Those bigram probabilities are estimated by counting occurrences over a training corpus. The main drawback of the relative frequency estimator is that it provides a null probability for events (*e.g.* succession of two words) never occurring in the training corpus. In order to overcome the sparse data problem, classifying words offers an alternative to smoothing techniques.
As the probability of a word given its predecessor, in a class-based language model, depends on its class and on its predecessor's class, the reduced number of classes implies a reduction of the amount of unseen events. Words that seldom occur in the training corpus are also better modeled as they can belong to a class which represents many occurrences.

A possible approach to establish a classification consists in automatically generating a partition of the vocabulary. In this case, multiple membership for a word is not allowed. Then, if $g(w)$ denotes the class of w, the probability of a word w_i given its predecessor w_{i-1} is computed as follows:

$$p(w_i \mid w_{i-1}) = p(w_i \mid g(w_i))\ p(g(w_i) \mid g(w_{i-1})) \qquad (1)$$

As it is computationally expensive to try every partition of a vocabulary as soon as its size gets large enough, greedy clustering algorithms have been proposed [1], [4].
In [1], Brown presents a bottom-up hierarchical clustering algorithm that proceeds as follows: initially, each word is assigned to its own class. Then, pairs of classes are successively merged, according to a criterion relying on the average mutual information between adjacent classes, until all the elements are grouped into a single cluster. At each iteration, the merged pair of classes is the one for which the resulting loss in average mutual information (computed over the training corpus) is the lowest.
The resulting hierarchical classification can be visualised through a binary tree, the root of which is the final single cluster and the leaves of which are the initial elements to be classified. Such a tree is called a dendogram [2]. Once the hierarchy is built, one has to extract a mapping of the words into classes. This is achieved by cutting the tree at a level given by the chosen final number of classes.

The criterion value is computed for each pair of classes and for each iteration. Given that V iterations are necessary to build the hierarchical classification tree for a vocabulary of V words, the complexity of the algorithm (which is mainly related to the updating of the distance matrix) increases dramatically as the size of the vocabulary grows. As a consequence, we have searched for a better trade-off between computational cost and language model performance by reducing the computational cost while keeping equivalent performance.
We then propose to improve this algorithm, in terms of computational cost on one hand, and in terms of speech recognition performance on the other hand. The principle is to reduce the cost of the distance matrix updatings by merging in a same iteration several pairs of classes. The next section presents the reciprocal neighbours algorithm and its application to the mutual information criterion.

2.THE RECIPROCAL NEIGHBOURS ALGORITHM

The objective of this section is to propose an alternative bottom-up hierarchical clustering algorithm relying on the notion of reciprocal neighbours [3]. This algorithm

allows one to reduce the number of iterations necessary to build a dendogram. Even if the complexity of each iteration is increased, the overall computational time is reduced and this is achieved without degrading performance.

2.1 Principles of the algorithm

Assume that we are dealing with a clustering task in which the criterion is a distance measure between two classes *i* and *j*: *d(i,j)*. If *G* is the current partition of the initial set, we define for any class *i* in *G*:

$$dd(i) = \min_{j \in G,\, j \neq i} \{d(i,j)\}$$

$$v(i) = \{j;\ j \in G,\ j \neq i;\ d(i,j) = dd(i)\}$$

Then two classes are reciprocal neighbours if $i \in v(j)$ and $j \in v(i)$. In other words, a pair of reciprocal neighbours is a pair of classes which are mutually nearest neighbours.

The clustering algorithm based on reciprocal neighbours proceeds as follows:

- at a given iteration, all the pairs of reciprocal neighbours are merged simultaneously[1].

As several merges are achieved in a single step, the main interest of this method is the reduction of the number of iterations. Although the complexity of each iteration is higher, the computational cost is also reduced because the global number of operations for distance matrix updating decreases.

From a theoretical point of view, the algorithm based on reciprocal neighbours leads exactly to the same classification as the basic one, providing that the distance measure *d* satisfies the reducibility axiom (also called the median axiom) [2]. Let {*h,h'*} denote the class obtained by merging the two classes *h* and *h'*.

Median axiom:
For any elements *h, h', h''* of a partition *G*,
if $d(h,h') \leq \min\{d(h,h''), d(h',h'')\}$
then $\min\{d(h,h''), d(h',h'')\} \leq d(\{h,h'\},h'')$.

In other words, if *d* is a distance measure satisfying this axiom. Assuming that (*h,h'*) and (*h'',h'''*) are two pairs of reciprocal neighbours for that distance, if we choose to merge first *h* and *h'* then the new class {*h,h'*} cannot be closer to *h''* than *h'''* is. Therefore, *h''* and *h'''* remain reciprocal neighbours after the merging. Hence, if the median axiom holds for the distance then both pairs may be merged in the same step leading to an equivalent result.

[1] In the case of three equidistant classes, a pair is chosen while the third class is left for further iterations.

2.2 Application of the algorithm to the loss of mutual information metric

Concerning our study, the distance is, as for the Brown algorithm, the loss of average mutual information. Let $p_k(l,m)$ denote the probability of observing in the training corpus the succession of two words from class *l* and class *m*, respectively, at iteration *k*. Let then $p_k(l)$ denote the probability of class *l* at iteration *k*.

$$p_k(l) = \sum_m p_k(l,m) = \sum_m p_k(m,l) \qquad (2)$$

The average mutual information between adjacent classes is defined at iteration *k* as:

$$I_k = \sum_{l,m} p_k(l,m) \log \frac{p_k(l,m)}{p_k(l)\, p_k(m)} \qquad (3)$$

The distance between two classes is then the difference between I_k and the resulting I_{k+1} that would be computed at the next iteration if those two classes were merged.

Given that merging two classes affects the loss of mutual information value for every other pair of classes, the median axiom, presented above, is not satisfied. The reciprocal neighbours clustering algorithm doesn't provide exactly the same classification tree as the previous bottom-up hierarchical one. Consequently, we have had to adapt the reciprocal neighbours algorithm to this particular metric.

Actually, it is possible that a merged class at a given iteration modifies the set of reciprocal neighbours determined for this same iteration. To be more precise, if two classes *h* and *h'* are reciprocal neighbours but are far from each other, it is more likely to observe that a newly formed class becomes closer to *h'* than *h* is. One possibility arises then, that is to set a threshold over the loss of mutual information value.

In order to choose appropriate thresholds, histograms of the values of the distance between selected reciprocal neighbours at each iteration have been plotted. From this observation, different ways of setting a threshold are proposed.

The first proposal is directly related to the histograms as the threshold is chosen to depend on both the minimal and the maximal value of the distance, among selected reciprocal neighbours.
Let RN_i denote the set of all reciprocal neighbours at iteration *i*. We define:

$$d_{min}^{\ i} = \min\{d(h,h');\ (h,h') \in RN_i\}$$
$$d_{max}^{\ i} = \max\{d(h,h');\ (h,h') \in RN_i\}$$

The threshold is computed according to the following formula, where *n* denotes a constant coefficient:

$$\Theta_i = d_{min}^{\ i} + \frac{d_{max}^{\ i} - d_{min}^{\ i}}{n} \qquad (4)$$

The set of neighbours effectively merged at iteration *i* is then:

$$RNT_i = \{ (h,h') \in RN_i / d(h,h') \leq \Theta_i \}$$

The second approach is easier to implement but allows us to study another phenomenon related to the first merges in the classification process. It consists in determining the minimal value of the distance for pairs in RN_i (as was done above) and in computing the threshold as follows:

$$T_i = \tau * d_{min}^{\ i} \qquad (5)$$

where τ is a constant real coefficient.

The problem with this method is that, for the first merges, the value of $d_{min}^{\ i}$ is very low and often unsignificant. Multiplying it by a coefficient is problematic. In order to overcome this problem, we set a fixed value T_0 for the threshold when $d_{min}^{\ i}$ is considered to be too low (below 10^5 in our experiments).

2.3 Low-occurring words

As we noticed above, bottom-up hierarchical clustering algorithms based on mutual information suffer from the fact that the first merges are decided upon using very low, and sometimes unsignificant, values of the criterion. A possible way to overcome this drawback is to treat separately words that seldomly occur in the training corpus. In fact, the membership of such words to one or another class has only little impact on the overall mutual information. It is easy to notice that most of the first merges concern low-occurring words.

We propose here a simple way to treat those words that is to cluster them all together into a single class which remains unchanged during the clustering process. If *MinCount* denotes the minimum number of occurrences for a word to be taken into account in the clustering task, the classification tree will be built only over those words occurring more than *MinCount* times as will be discussed in the next section.

3. EXPERIMENTAL RESULTS

Experiments are run on a spontaneous speech corpus composed of transcriptions of human-computer spoken dialogues on the AGS voice service directory inquiry demonstrator [5]. The vocabulary contains 858 different words, 782 of them occurring in the training text. The training corpus consists of 26362 words (5661 sentences), and the test corpus consists of 4041 words from 816 sentences.

In order to compute perplexity on the test set for the different models, probabilities are smoothed according to a smoothing threshold. Unseen events probabilities are set to a constant small value ε.

Speech recognition tests are carried out with a HMM-based, speaker independent, continuous speech recognition software working over the telephone network.

First of all, the class-based model constructed with the Brown algorithm classification is tested and compared with a word bigram model. A minimum for both perplexity and word error rate is reached at approximately 300 classes. At that point, both are below the corresponding perplexity or word error of the word bigram model rate (4% reduction for the word error rate).

Concerning the reciprocal neighbours algorithm, the classification tree is built in far less iterations. Although each iteration has a higher complexity, the CPU time needed to build the tree is lowered. The following table indicates both number of iterations and CPU time for different algorithms (notations are defined in the previous section):

Tab.1: Iterations, CPU time and CPU reduction for various algorithms.

	Number of iterations	CPU time (s)	CPU reduction
Brown	782	1864	
R.N. no threshold	116	1629	13 %
R.N. $n = 2$	152	1584	15 %
R.N. $\tau = 5$; $T_0 = 1.5$	113	1629	13 %
Brown *MinCount* = 3	468	468	
R.N. *MinCount* = 3	85	424	9 %

Table 1 indicates a reduction of the number of iterations by a factor from around 5 to 7, while the CPU time is reduced by 13% to 15%.

One of the main advantages of the reciprocal neighbours algorithm is that it provides a classification tree with different levels, each level corresponding to one iteration. All the neighbours that are merged in a given iteration belong to the same level. Level 0 corresponds to the initialisation of the algorithm where each word belongs to its own class. The root of the tree, where all words are gathered into a single class is the highest level. A mapping can then be extracted from the tree for every level.

Perplexity as well as word error rate are evaluated on the test corpus.

The first results compare the Brown algorithm and reciprocal neighbours algorithm in its original form (that is without any threshold). Figures 1 and 2 show that both outperform the word bigram model and the

reciprocal neighbours algorithm gives slightly better results.

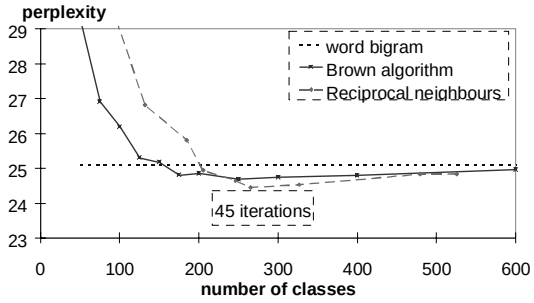

Fig. 1: Perplexity for different numbers of classes

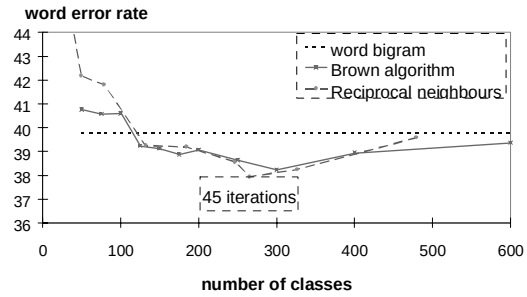

Fig. 2: Word error rate for different numbers of classes

Even though the latter seems to be less precise, it is possible to explain why it gives better results when considering the first merges. As we said previously, the first values of the criterion are often unsignificant and the choice of one pair of classes to merge rather than another one is not very reliable. On the contrary, with the reciprocal neighbours algorithms many merges are done in the very first iterations. If the average number of merges per iteration is 5.14, there are 83 merges in the first iteration and, at the fifth iteration, a total of 166 pairs of classes are merged. Thus, perturbations due to numerical unaccuracies are reduced.

We now compare different versions of the reciprocal neighbours algorithm, that is different ways of setting a threshold over the loss of mutual information values.
Considering the first possibility derived from the histograms, testing several values for *n* have shown that 2 was the optimal value. It performs equivalently with a slight reduction in CPU time.
As for the second possibility, the optimal values have proven to be τ=5 for the multiplicative coefficient and T_0 = 1.5 for the fixed threshold applied to low values of the minimal distance.
Both lead to equivalent performance as the original algorithm. This allows to conclude that even if the median axiom is not satisfied and if the hierarchical algorithm based on reciprocal neighbours doesn't provide the same dendogram as the simple hierarchical one, it doesn't affect the performance and actually gives slightly better results.

The last set of experiments deals with low occurring words. Several values of the minimum number of occurrences for a word to be taken into account during the clustering process (*MinCount*) have been tested. For *MinCount* = 3, the number of clustered words drops from 782 to 468. The tree is then built in far less time, as indicated in the table at the beginning of this section.
Here again, the reciprocal neighbours algorithm applied to words occurring more than *MinCount* times in the training corpus is faster than the Brown algorithm and the results are slightly better with the reciprocal neighbours algorithm.
It is interesting to note that not classifying the low-occurring words lead to a 4-fold reduction in computational cost while keeping performances equivalent to the case where all the words are clustered.

4.CONCLUSION

A new bottom-up hierarchical clustering algorithm is presented which enables a large reduction of the number of iterations necessary to build a classification tree as well as the corresponding CPU time. The speech recognition word error rate is also reduced. A possible interpretation is that the algorithm minimises the effects of numerical unaccuracy in computing the low values of the metric during the first merges of the clustering process. Further tests on larger corpora should help us in the interpretation.
Another interest of this hierarchical clustering algorithm is that it provides a tree where each level corresponds to a specific classification. In future work, we will investigate extracting an optimal level of classification for each word of the vocabulary.

REFERENCES

[1] P.F. Brown, V.J. Della Pietra, P.V. deSouza, J.C. Lai, R.L. Mercer: « *Class-Based n-gram Models of Natural Language* ». Computational Linguistics, 18(4), pp. 467-479, 1992.

[2] G. Celeux, E. Diday, G. Govaert, Y. Lechevallier, H. Ralambondrainy: « *Classification automatique des données* », (in French) Dunod informatique, 1989.

[3] C. de Rham: « *La classification hiérarchique ascendante selon la méthode des voisins réciproques* », (in French) Les Cahiers de l'Analyse des Données, vol. V, 2, pp. 135-144, 1980.

[4] R. Kneser, H. Ney: « *Improved clustering techniques for class-based statistical language modelling* », Proc. EUROSPEECH'93, Berlin, pp. 973-976, 1993.

[5] D. Sadek, A. Ferrieux, A. Cozannet, P. Bretier, F. Panaget, J. Simonin: « *Effective Human-Computer Cooperative Spoken Dialogue: The AGS Demonstrator* », Proc. ICSLP'96, Philadelphia, USA, 1996.