Incremental and iterative monolingual clustering algorithms

Sergio Barrachina

Dpt. of Computer Engineering
and Science
Universidad Jaume I
barrachi@icc.uji.es

Juan Miguel Vilar

Dpt. of Programming Languages
and Computer Systems
Universidad Jaume I
jvilar@lsi.uji.es

Abstract

To reduce speech recognition error rate we can use better statistical language models. These models can be improved by grouping words into word equivalence classes. Clustering algorithms can be used to automatically do this word grouping.

We present an incremental clustering algorithm and two iterative clustering algorithms. Also, we compare them with previous algorithms.

The experimental results show that the two iterative algorithms perform as well as previous ones. It should be pointed out that one of them, that uses the leaving one out technique, has the ability to automatically determine the optimum number of classes. These iterative algorithms are used by the incremental one.

On the other hand, the proposed incremental algorithm achieves the best results of the compared algorithms, its behavior is the most regular with the variation of the number of classes and can automatically determine the optimum number of classes.

1. Introduction

The need for a statistical language model in speech recognition arises from Bayes' decision rule for minimum error rate. The word sequence $w_1 \dots w_N$ to be recognized from the sequence of acoustic observations $x_1 \dots x_T$ is determined as that word sequence $w_1 \dots w_N$ for which the posterior probability $\mathrm{P}(w_1 \dots w_N \mid x_1 \dots x_T)$ attains its maximum. This rule can be rewritten in the form:

$$\arg\max_{w_1 \cdots w_N} \mathrm{P}(w_1 \dots w_N)\mathrm{P}(x_1 \dots x_T \mid w_1 \dots w_N),$$

where $\mathrm{P}(x_1 \dots x_T \mid w_1 \dots w_N)$ is the conditional probability of, given the word sequence $w_1 \dots w_N$, observing the sequence of acoustic measurements $x_1 \dots x_T$ and where $\mathrm{P}(w_1 \dots w_N)$ is the prior probability of producing the word sequence $w_1 \dots w_N$.

The task of a statistical language model is to provide estimates of the prior probability $\mathrm{P}(w_1 \dots w_N)$. This is usually computed as the product of a sequence of conditional probabilities $\mathrm{P}(w_i \mid w_1 \dots w_{i-1})$. This product, when using a bigram model, is approximated by $\prod_{i=1}^{N} \mathrm{P}(w_i \mid w_{i-1})$.

The bigram model is estimated from a text corpus during a training phase. The main problem we are faced with is that most of the possible events, i.e. word pairs, are never seen in training because there are so many of them. To overcome this problem we could try to cluster similar words into word equivalence classes. If we can successfully assign words to classes, it may be possible to make more reasonable predictions for histories that we have not previously seen by assuming that they are similar to other histories that we have seen [1]. Therefore, we can make a better estimation of the language model.

The rest of this paper is organized as follows. In section 2 we will show some basic concepts of clustering and some previous clustering algorithms. In section 3 we will present the proposed clustering algorithms. In section 4 we will show experiments with the described clustering algorithms. Lastly, in section 5, we will present the conclusions and future work.

This work has been partially funded by the European Union's TransType 2 project (n. IST-2001-32091) and the *Bancaja*'s SIEsTA project (n. P1·1B2002-15)

2. Basic Concepts

The aim of a clustering algorithm is to find the best possible mapping of words to classes. To find the best clustering, it seems obvious that, given the necessary time and resources, the best way to assign each word to its class will be to enumerate all the possible partitions of words in the desired number of classes and to select the partition that maximizes a given optimization function. However, this approach can not be followed, even for a moderated number of classes and words, due to the huge number of possible partitions.

Due to this limitation, clustering algorithms restrict themselves to evaluate just a small fraction of the possible partitions. As one of these partitions will be selected as the best one, the proposed solution will be suboptimal. Therefore, the results obtained by a clustering algorithm will depend on the criteria used to select which partitions should be considered.

Two of the most known clustering algorithms in the language modeling field were proposed in 1992 by Brown et al. [1]. These methods do a hard clustering of words into classes. They define a mapping function $\mathrm{G} : w \to g$ that assigns each word to a given class. Given this function, the conditional probability of seeing a word w_2 given the word w_1 can be expressed as:

$$\mathrm{P}(w_2 \mid w_1) = \mathrm{P}(w_2 \mid \mathrm{G}(w_2))\mathrm{P}(\mathrm{G}(w_2) \mid \mathrm{G}(w_1)) \quad (1)$$

Given a training corpus and the function G, the probabilities $\mathrm{P}(w_2 \mid \mathrm{G}(w_2))$ and $\mathrm{P}(\mathrm{G}(w_2) \mid \mathrm{G}(w_1))$ can be simply estimated counting the number of times that the appropriate events occur. Therefore, only the function G must be determined. Brown et al. [1] propose to determine this function using as optimization function the reduction of the perplexity of the class based language model.

The first of the algorithms proposed by Brown et al. [1] is a hierarchical agglomerative clustering algorithm. All the words are initially assigned to a different class. The algorithm proceeds by merging two classes at each turn until the desired number of classes is reached. In each iteration, all possible merges are evaluated and the one which gives the best improvement on the perplexity is carried out. Lastly, to compensate early mergings, words are moved to other classes seeking to improve the perplexity. This algorithm has a $O(V^3)$ time complexity, where V is the size of the vocabulary.

The second algorithm proposed by Brown et al. [1] has a time complexity of $O(V \cdot C^2)$, where C is the number of desired classes. This algorithm begins sorting all the words in the training corpus by their number of occurrences and assigning each of the C most frequent words to a different class. Then, the next word is assigned to a new class. The best merging of two of the $C + 1$ classes is searched and carried out; leading again to C classes. This process is repeated for each remaining word.

Another clustering method broadly used in the language modeling field is the one of Kneser and Ney [2]. This algorithm instead of beginning with as much classes as words, begins with an initial mapping of all the words in the C desired classes. What they do to search a better clustering is to move a word from its current class to another one. The time complexity of each iteration of this algorithm is $O(N + V \cdot C^2)$, where N is the number of words in the training corpus.

Martin et al. [3] improve the algorithm proposed by Kneser and Ney [2] storing the count of each pair of observed words. This way, instead of visiting each occurrence of the word w in the training corpus, all the bigrams in which w occurs are collected. If these counts are stored in a direct access matrix, then the time complexity for each iteration is $O(B + V \cdot C^2)$, where B is the number of seen bigrams in the training corpus.

When the vocabulary size is large, direct access becomes prohibitive due to the large memory requirements. Therefore, Martin et al. [3] propose the use of lists and binary search to store the bigram counts. Since there are B/V bigrams per word on the average, the resulting time complexity is $O(B \cdot \log(\frac{B}{V}) + V \cdot C^2)$.

3. Clustering algorithms

The first two clustering algorithms proposed in this paper are based on the ones of Kneser and Ney [2] using the optimizations described in Martin et al. [3]. The third algorithm proposed is able to carry out the clustering incrementally; it begins with one class and generates newer classes until the desired number of classes is reached.

As optimization criterion we have used the reduction of the perplexity. This is equivalent to the increase of the mutual information between adjacent classes, $\mathrm{I}(c_1; c_2)$ [1]. Therefore, the optimum clustering function, $\mathrm{G}_{\mathrm{opt}}$, is the one that maximizes $\mathrm{I}(c_1; c_2)$:

$$\mathrm{G}_{\mathrm{opt}} = \arg\max_{\mathrm{G}} \sum_{c_1 c_2} \mathrm{P}(c_1 c_2) \log \frac{\mathrm{P}(c_1 c_2)}{\mathrm{P}(c_1)\mathrm{P}(c_2)}. \qquad (2)$$

3.1. Iterative clustering algorithm

This algorithm (`iter`) proceeds as follows (see algorithm 1). Firstly, an initial distribution of the words into the C classes is chosen. Then, one of the words w is tentatively moved from its class $\mathrm{G}(w)$ to the other classes. For each tentative movement, the variation of $\mathrm{I}(c_1; c_2)$ is computed. Once all of the tentative movements have been evaluated, the movement which most increases $\mathrm{I}(c_1; c_2)$ is carried out. The same process is repeated for each word. When all the words have been tried another iteration starts unless no word has been moved or a fixed number of iterations is reached. It should be pointed out that the words are evaluated in frequency order, first the most frequent ones. Another factor to take into account is that a word can not be removed from its class when it is the only one in that class.

Algorithm 1 Iterative clustering

```
Require: $C$, $maxIter$
  Initialize G
  repeat
    for all $w$ in descending frequency order do
      Tentatively move $w$ to each class $c$
      Move $w$ to the class $c$ that most increases $\mathrm{I}(c_1; c_2)$
    end for
  until $maxIter$ is reached o no word has been moved
```

The initial mapping of words to classes chosen in [3] assigns the most frequent words to the first $C - 1$ classes and the remainder words to the last class. If this initial distribution is used, the most frequent words can not be moved until some other words are assigned to their class (a word can not be moved if it is the only one in its class). This initial mapping does not allow the most frequent words, i.e. the words we know more about, to take part in the early stages of the clustering process.

Instead of using this initial mapping, we assign all the words but the less frequent $C - 1$ words to one class, and the less frequent $C - 1$ words each to one class. This way, the most frequent words can be moved from the very beginning, what we think will lead to a better clustering [4].

The objective function, $\mathrm{I}(c_1; c_2)$, is estimated as:

$$\mathrm{I}(c_1; c_2) \simeq \frac{1}{N-1} \left[\sum_{c_1 c_2} \mathrm{n}(c_1 c_2) \cdot \log \frac{\mathrm{n}(c_1 c_2)}{N-1} - 2 \sum_{c} \mathrm{n}(c) \cdot \log \frac{\mathrm{n}(c)}{N-1} \right], \qquad (3)$$

where N is the number of words and $\mathrm{n}(\cdot)$ is the number of times an event occurs in the corpus.

It should be noted that is not necessary to recompute $\mathrm{I}(c_1; c_2)$ for each movement [2]; we can compute just the variation due to the current movement by updating the affected counts in (3). In addition to this and to the optimization proposed in [3] of storing the bigram counts, the following optimizations have been made:

- $\Delta\mathrm{I}(c_1; c_2)$ is computed in two steps: firstly, the variation due to the extraction of the word w from its class and secondly, the variation due to the insertion of the word w in the destination class. This way, the first step is computed just once for each word under consideration.
- The classes that precede and follow the word which is going to be moved, w, are identified in advance, as just the counts related to these classes will be modified by the movement of word w. This identification is made once each word is under consideration.

The variation of $\mathrm{I}(c_1; c_2)$ is easily computed: the original contribution of the modified counts is removed from $\mathrm{I}(c_1; c_2)$ and the new contribution is added. We just need to know how the counts $\mathrm{n}(c_1 c_2)$ and $\mathrm{n}(c)$ are modified when a word is moved. When a word w is extracted from its class c_w, the new counts become:

$$\mathrm{n}'(c_1c_2) = \begin{cases} \mathrm{n}(c_1c_2) - \mathrm{n}(wc_2) & \text{if } c_1 = c_w \wedge c_2 \neq c_w \\ \mathrm{n}(c_1c_2) - \mathrm{n}(c_1w) & \text{if } c_1 \neq c_w \wedge c_2 = c_w \\ \mathrm{n}(c_1c_2) + \mathrm{n}(ww) & \\ \quad - \mathrm{n}(wc_w) & \\ \quad - \mathrm{n}(c_ww) & \text{if } c_1 = c_2 = c_w \\ \mathrm{n}(c_1c_2) & \text{in other case} \end{cases} \tag{4}$$

$$\mathrm{n}'(c) = \begin{cases} \mathrm{n}(c) - \mathrm{n}(w) & \text{if } c = c_w \\ \mathrm{n}(c) & \text{in other case} \end{cases} \tag{5}$$

Using these new counts we can compute the variation of $\mathrm{I}(c_1; c_2)$ due to the extraction of word w from c_w. We call this variation, $\Delta\mathrm{I}(c_1; c_2)|_{ext}$. Similarly, when the word w is added to the class c_d new counts $\mathrm{n}''(c_1c_2)$ and $\mathrm{n}''(c)$ can be obtained. Using the $\mathrm{n}''(\cdot)$ and $\mathrm{n}'(\cdot)$ counts we can compute the variation of $\mathrm{I}(c_1; c_2)$ due to the insertion of word w into the class c_w. We call this variation, $\Delta\mathrm{I}(c_1; c_2)|_{ins}$. Finally, the variation due to the movement of the word w from the class c_w to the class c_d can be obtained as:

$$\Delta\mathrm{I}(c_1; c_2) = \Delta\mathrm{I}(c_1; c_2)|_{ext} + \Delta\mathrm{I}(c_1; c_2)|_{ins}.$$

3.2. Iterative leaving one out clustering algorithm

This algorithm (`iter_lo`), as the preceding one, follows the steps shown in algorithm 1. The difference with the previous algorithm is that it employs the leaving one out method [2] to compute the objective function $\mathrm{I}(c_1; c_2)$.

The leaving one out method is a special type of cross validation where the training text is divided into $N - 1$ samples as the *retained* part, and 1 sample as the *held-out* part. This process is repeated N times so that all N partitions are considered. It is used to simulate unseen events and to avoid the overfitting of the model to the training data.

Using the leaving one out method and the absolute discounting smoothing technique, $\mathrm{I}_{lo}(c_1; c_2)$ is estimated as:

$$\begin{aligned} \mathrm{I}_{lo}(c_1; c_2) \simeq & \sum_{c_1c_2:\mathrm{n}(c_1c_2)>1} \frac{\mathrm{n}(c_1c_2)}{N-1} \log \frac{\mathrm{n}(c_1c_2) - 1 - b}{N-2} \\ & + \frac{n_1}{N-1} \log \frac{(n_+ - 1)b}{(n_0 + 1)(N-2)} \\ & - 2 \sum_{c} \frac{\mathrm{n}(c)}{N-1} \log \frac{\mathrm{n}(c) - 1}{N-2}, \end{aligned} \tag{6}$$

where b is the absolute discounting constant and n_0, n_1, and n_+ are the number of events c_1c_2 that has been seen zero, one or more than one times, respectively. The b parameter can be estimated as $n_1/(n_1 + 2n_2)$ [5, pg. 189], where n_2 is the number of events c_1c_2 that has been seen twice in the training corpus.

The optimization techniques described for the `iter` algorithm can also be applied to this one. Nevertheless, the counts of counts n_0, n_1, and n_+, in addition to the $\mathrm{n}(c_1c_2)$ and $\mathrm{n}(c)$ counts, have to be updated with each tentative movement.

3.3. Incremental leaving one out clustering algorithm

This algorithm (`inc_lo`, see algorithm 2) tries to overcome a limitation of the iterative algorithms: different initial mappings lead to different final clusterings.

Trying different initial mappings to select the best final clustering will just consider a few number of all the possible initial partitions. Instead of this approach, we propose the next strategy: to cluster the words in C classes using a previously obtained clustering with $C - 1$ classes.

Algorithm 2 Incremental leaving one out clustering

```
Require: $C$, $maxIter$
  Initialize G : $\mathrm{G}(w_i) \leftarrow 1, \forall w_i; n \leftarrow 1$
  repeat
    Create a new empty class $c_{n+1}$
    Tentatively move each word $w$ to the new class $c_{n+1}$
    Move the word $w_i$ that most improves $\mathrm{I}(c_1; c_2)$ to $c_{n+1}$
    if there is such a word $w_i$ then
      for all $w_j$ in the previous class of $w_i$, $c_i$, do
        Move $w_j$ to $c_{n+1}$ if $\mathrm{I}(c_1; c_2)$ is increased
      end for
      $n \leftarrow n + 1$
      Do iter_lo($n, maxIter$)
    end if
  until $n = C$ or $w_i = \emptyset$
```

If we have a clustering with $C - 1$ classes we can easily obtain an initial mapping for C classes. We can begin creating a new class and moving to it the word whose movement will most improve $\mathrm{I}_{lo}(c_1; c_2)$. Then, the remaining words in the previous class of the moved word are tried in turn. If their movement will improve the objective function, they are moved. Once we have this initial mapping we apply the `iter_lo` algorithm. (We could also use $\mathrm{I}(c_1; c_2)$ as objective function and `iter` as clustering algorithm.)

4. Experimental results

For the experiments we have used the EuTrans I corpus. This is an Spanish–English bilingual corpus with 10,000 training sentence pairs and 2,996 test sentence pairs. It is a limited domain corpus intended for machine translation purposes. It covers some communication situations between a client and a receptionist of an hotel. It has a vocabulary of 686 Spanish words and 513 English words [6]. Although we only show in this paper the results obtained with the Spanish part of this corpus, similar results have been achieved with the English part.

We have evaluated, in addition to the proposed clustering algorithms, the following ones: the Brown et al. [1] $O(V \cdot C^2)$ algorithm (`srilm_inc`), the Brown et al. [1] $O(V^3)$ algorithm (`srilm_full`) and the algorithm described in Och [7] based on [2, 3] (mkcls). The `ngram-class`[1] and `mkcls`[2] applications have been used to generate these clusterings.

To compare the quality of the clusterings obtained by each method we have obtained a class bigram language model for each clustering. Then we have used each of these models to compute the perplexity of the test. The lower the perplexity, the better the model. To obtain the language models and to measure the perplexities we have used the `ngram` application from the *SRI Language Modeling Toolkit*.

For each clustering method, we have clustered the training corpus with different number of classes between 100 and 686 (the size of the vocabulary). The perplexities obtained are shown in figure 1. As it can been seen, the best result is achieved with 250 classes by the `inc_lo` clustering method. It should also be pointed out that the perplexity of the other methods quickly increases when the number of classes goes beyond 250 while the perplexity of the `inc_lo` method has a more smooth

[1]http://www.speech.sri.com/projects/srilm/

[2]http://www-i6.informatik.rwth-aachen.de/˜och/

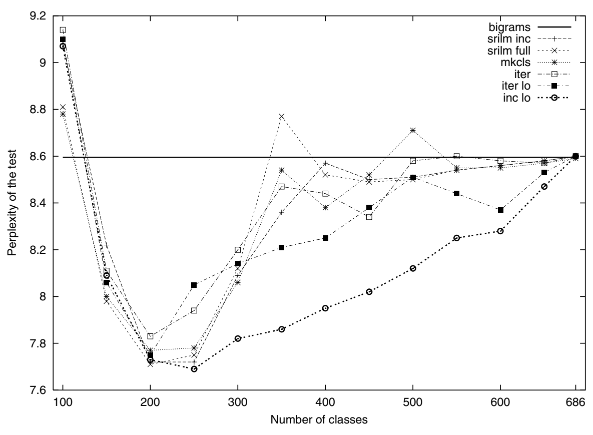

Figure 1: *Perplexity measured on the test for different class-based model languages.*

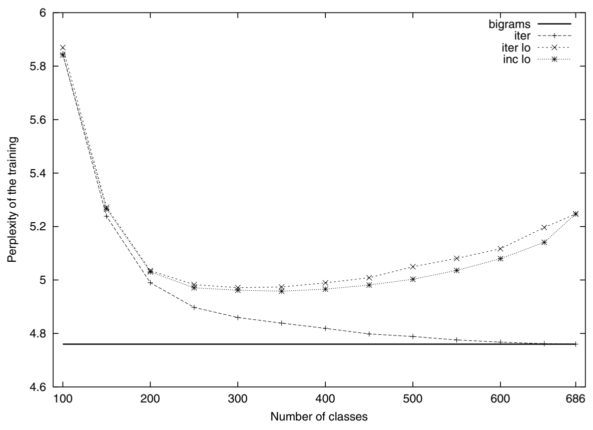

Figure 2: *Perplexity measured on the training for different class-based model languages.*

behavior with the variation of the number of classes. It can also be seen that the other methods obtain similar perplexities. Therefore, we expect the clusterings obtained by these methods to be quite similar.

With the next experiment we wanted to check if the leaving one out methods can automatically determine the optimum number of classes. To carry out this experiment we have clustered the training part of the corpus with the `iter`, `iter_lo` and `inc_lo` algorithms for different number of classes between 100 and 686. For each number of classes we have obtained the perplexity of the training part as is computed by each method.

The results of this experiment are shown in figure 2. As it can be seen, the perplexity obtained by the `iter` method decreases as the number of classes increases, while the perplexity given by the `iter_lo` and `inc_lo` methods have a minimum in 300 and 350 classes, respectively. We would like to use these minima as an indication of the optimum number of classes. As we have seen in figure 1, the best number of classes, according to the perplexity of the test, was 250; which is relatively near to the optimum ones determined automatically. Furthermore, the difference between the perplexities obtained by the `inc_lo` method for 250 and 350 class is small. Therefore, the `iter_lo` and `inc_lo` algorithms provide a good approximation of the optimum number of classes automatically.

5. Conclusions and future work

We have proposed in this paper an incremental algorithm to automatically cluster words in equivalence classes. This algorithm begins with just one class and increments the number of classes until the desired number is reached. For each new number of classes an iterative type clustering algorithm is used to obtain a new clustering. For this purpose, we have also proposed two iterative clustering algorithms.

As the main objective of the clustering is to obtain better language models that could in turn lead to better speech recognition rates, we have compared the perplexity obtained by the clustering methods proposed and the ones described in [1, 2, 3].

The experimental results show that the proposed incremental algorithm obtains better results than the other ones. It should also be pointed out that this algorithm has the most regular behaviour when the number of classes changes. It is also shown that the proposed iterative clustering algorithms perform as well as previous clustering algorithms.

Finally, we would like to extend the proposed algorithms in order to do asymmetric clustering. The assymetric clustering uses different clusters for both predicted and conditional words and seems to obtain better results than classical models [8].

6. References

[1] 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, pp. 467–479, 1992.

[2] R. Kneser and H. Ney, "Improved clustering techniques for class-based statistical language modelling," in *proceedings of the EuroSpeech'93*, vol. 2, (Berlin (Germany)), pp. 973–976, 1993.

[3] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," in *Proceedings of the EuroSpeech'95*, (Madrid (Spain)), pp. 1253–1256, 1995.

[4] S. Barrachina and J. Vilar, "Improved bilingual clustering through monolingual clustering," in *Proceedings of the CAEPIA'99*, vol. 1, (Murcia (Spain)), pp. 183–190, Nov. 1999.

[5] S. Young and G. Bloothooft, eds., *Corpus-based Methods in Language and Speech Processing*. Kluwer Academic, 1997.

[6] J. C. Amengual, J. M. Benedí, F. Casacuberta, A. Castano, A. Castellanos, D. Llorens, A. Marzal, F. Prat, E. Vidal, and J. M. Vilar, "Using categories in the EuTrans system," in *Proceedings of the Spoken Language Translation Workshop, ACL and European Network in Language and Speech*, (Madrid, España), pp. 44–53, 1997.

[7] F. Och, "An efficient method for determining bilingual word classes," in *Proceedings of the Ninth Conf. of the Europ. Chapter of the Association for Computational Linguistics, EACL'99*, (Bergen (Norway)), pp. 71–76, June 1999.

[8] J. Gao, J. T. Goodman, G. Cao, and H. Li, "Exploring asymmetric clustering for statistical language modeling," in *Proceddings of the 40th Annual Meeting of te Association for Computational Linguistics (ACL)*, (Philadelphia), pp. 183–190, July 2002.