Clustered Language Models with Context-Equivalent States

J.P. Ueberla * *and I.R. Gransden* **

* Forum Technology ** Speech Research Unit
DRA Malvern, St.Andrews Road
Malvern, Worcestershire, WR14 3PS, UK
email:{ueberla,gransden}@signal.dra.hmg.gb

ABSTRACT

In this paper, a hierarchical context definition is added to an existing clustering algorithm in order to increase its robustness. The resulting algorithm, which clusters contexts and events separately, is used to experiment with different ways of defining the context a language model takes into account. The contexts range from standard bigram and trigram contexts to part of speech five-grams. Although none of the models can compete directly with a backoff trigram, they give up to 9% improvement in perplexity when interpolated with a trigram. Moreover, the modified version of the algorithm leads to a performance increase over the original version of up to 12%.

1. Introduction

The task of a language model is to calculate $p(w_i|c_i)$, the probability of the next word being w_i given the current context c_i. Language models differ in the way this probability is modelled and how the context c_i is defined. A quite general model proposed in [4] makes use of a state mapping function S and a category mapping function G. The idea behind the state mapping $S : c- > s_c = S(c)$ is to assign each of the large number of possible contexts $c \in C$ to one of a smaller number of context-equivalent states. Similarly, the category mapping $G : w- > g_w = G(w)$ assigns each of the large number of possible words $w \in V$ to one of a smaller number of categories (similar to parts of speech). The probability of the next word is then calculated as

$$p(w_i|c_i) = p(G(w_i)|S(c_i)) * p(w_i|G(w_i)). \qquad (1)$$

In [6], a heuristic version of a clustering algorithm was presented, which can be used to calculate S and G automatically. In this paper, the algorithm is extended to deal with a hierarchy of contexts, which increases its robustness (Section 2.). It is then used to experiment with different ways of defining the context, including the use of parts of speech information. The different models are evaluated in terms of perplexity on the Wall Street Journal Corpus (Section 4.).

2. Clustering Algorithm

The initial clustering algorithm used to determine S and G automatically is shown in Figure 1. It is a greedy, hill-

Algorithm 1: Clustering()

start with initial clustering functions S, G
iterate until some convergence criterion is met
for all $w \in V$ and $c \in C$
for all $g'_w \in G$ and $s'_c \in S$
calculate the difference in the optimisation criterion when w/c is moved from g_w/s_c to g'_w/s'_c
move the w/c to the g'_w/s'_c that results in the biggest improvement in optimisation criterion

End Clustering

Figure 1: The clustering algorithm

climbing algorithm that moves elements to the best available choice at any given time. For more details about the algorithm, the optimisation criterion and its heuristic version (which is used in all the experiments reported here), please refer to [6].

A major drawback of the algorithm becomes apparent when it is used for wider contexts. Since S clusters individual contexts, many of these contexts have occurred only infrequently in the training data. It is therefore very difficult to assign them to a meaningful cluster. In fact, the algorithm doesn't attempt to move elements which have occurred less than a minimal number of times (the empirically determined value of 6 was used for this threshold in our experiments). Depending on the number of elements for which this is true, this can lead to poor performance. In the trigram case, for example, 85% of the distinct contexts seen during training have occurred less than 6 times.

The main idea to improve upon this situation is as follows. Rather than moving individual contexts, the algorithm first moves groups of contexts together. Each group will have oc-

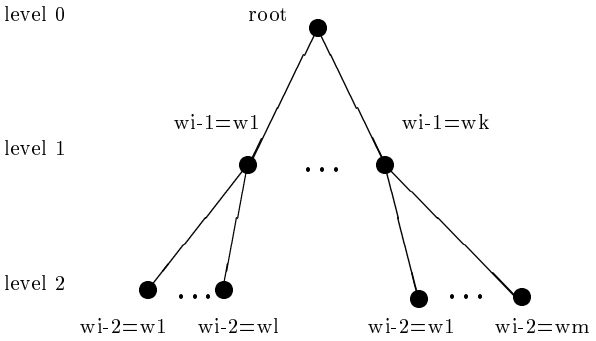

Figure 2: Example of a trigram tree

curred more frequently and hence its statistics will be more reliable. Only later on is the algorithm allowed to move individual contexts. As an example consider the trigram case, where the context is defined by the pair of previous words (w_{i-2}, w_{i-1}). Initially, the algorithm moves all contexts which have the same w_{i-1} together (e.g. identical bigram contexts). Subsequently, it proceeds by moving pairs of words.

In more general terms, we can represent the groupings of the contexts in terms of a tree T. The leaves of T correspond to all the different contexts seen during training. The nodes at each of the $0 \leq l \leq L-1$ levels of the tree correspond to a classification of all the contexts into a smaller set of groups. For example, the tree shown in Figure 2 corresponds to the trigram case, where context with the same w_{i-1} are grouped together.

It is quite simple to modify the clustering algorithm to make use of such a tree T. Let $N(T,l)$ denote the set of nodes of T at level l and let Contexts(n) denote the set of contexts below a node n (e.g. all the leaves dominated by n). The resulting clustering algorithm is shown in Figure 3.

Although a tree can be used to represent many different ways of grouping contexts, we have so far only experimented with very simple trees. Let each context c be defined by a L-tuple of values $c = (v_L, ..., v_l, ...v_1)$. The trees used in our experiments always group contexts which have identical subcontexts together. Thus, the ith level of the tree has one node for each existing i-tuple of values $(v_i, ..., v_1)$ and each such node contains all the contexts which are further refinements of this i-tuple.

3. Test Corpus and Clustering Times

Using the non-verbalised version of the Wall Street Journal corpus (approximately 38 million words, 20,000 word vocabulary), different language models were evaluated in terms of perplexity. We use the same conditions as [5] and [2] in order to make direct comparisons of perplexity possible.

All of the results described in this paper were obtained with-

Algorithm 2: Clustering()

start with initial clustering functions S, G
for each level l of tree T
iterate until some convergence criterion is met
for all $w \in V$ and $n \in N(T,l)$
for all $g'_w \in G$ and $s'_n \in S$
calculate the difference in the optimisation criterion when w is moved from g_w to g'_w or when all $c \in$ Contexts(n) are moved from s_n to s'_n
move the w to the g'_w or all $c \in$ Contexts(n) to the s'_n that result in the biggest improvement in the optimisation criterion

End Clustering

Figure 3: The tree-based clustering algorithm

clustered			
Context	Clusters (S,G)	PP	PP (tree)
w_{i-1}	500,500	242	-
w_{i-1}	2000,2000	190	-
w_{i-2}, w_{i-1}	2000,2000	180	158
backoff			
Context			PP
w_{i-1}			172
w_{i-2}, w_{i-1}			112

Table 1: Comparison of the clustering algorithm and standard n-grams

out putting much effort into individual parameter tuning. The threshold below which elements are not moved by the algorithm was set to 6 in all experiments. As convergence criterion, the relative improvement in the value of the optimisation function during the last iteration was used. If that improvement is less than 1%, no more iterations are performed. This results in only two iterations in most cases, which is significantly less than the about 20 to 30 iterations mentioned in [3]. Hence there is reason to believe that some of the results could be improved upon by better optimisation of these parameters.

Using the heuristic version of the algorithm presented in [6], one iteration in the bigram case takes about 5 hours (elapsed time, not CPU) on a DEC alpha workstation. The complete clustering takes about 10 hours for a bigram and 3 days for a trigram. The time required for most of the other models lies in between the bigram and trigram case.

4. Results

In a first set of experiments, the clustering algorithms were compared to the standard bigram and trigram models. The results are shown in Table 1. First, one can compare our

Context	Clusters (S,G)	PP	PP (tree)
t_{i-1}	2000,2000	443	-
t_{i-2}, t_{i-1}	2000,2000	343	342
$t_{i-3}, t_{i-2}, t_{i-1}$	2000,2000	305	301
$t_{i-4}, t_{i-3}, t_{i-2}, t_{i-1}$	2000,2000	305	292

Table 2: Using 61 linguistic parts of speech tags t

Context	Clusters (S,G)	PP	PP (tree)
g_{i-2}, g_{i-1}	2000,2000	184	170

Table 3: Using 1000 clustering classes g

backoff results to those reported in [2]. Our backoff bigram result is about 2%, the trigram result about 7% worse. The difference could be explained by the different smoothing technique we use and by the fact that our trigram discards singleton events. Second, one can see that the backoff models outperform the clustered models. This is especially true for the trigram. It is worth noting, however, that the trigram has approximately 14 million parameters, as compared to 4 million for the clustered model. Third, Table 1 also shows that the tree-based version of the algorithm outperforms the original one, giving an improvement of 12%. Finally, the clustered bigram results using 500 clusters allows a direct comparison with the one given in [3], where a very similar perplexity figure of 244 is given.

In a second set of experiments, the use of parts of speech information in the context definition was investigated. Due to limitations of our software, each word could belong to one part of speech only. Brill's rule based tagger [1] was therefore employed to assign the most likely tag t to each word in the official 20K vocabulary used in the language modeling experiments. This resulted in 61 different tags. Table 2 gives the results for various models using this part of speech information. As the size of the context window increases, the tree based version of the algorithm gives an increasing gain in performance . When moving from a window size of three to four, the standard version of the clustering algorithm does not lead to an improvement (by looking at one extra digit, one can see that it decreases from 304.6 to 305.0). This is presumably because of the data sparseness problem mentioned in Section 2. The performance of the tree based version, however, continues to increase.

In a third set of experiments, the clustering of words G produced by the algorithm was used to define the context. Compared to using the linguistic parts of speech, this has the advantage that the number of classes can be determined almost at will. The perplexities for a model that uses 1000 different classes are shown in Table 3. One can again see the benefit of using the tree based version. Moreover, it is interesting to note that the resulting perplexity comes quite close to that of a clustered trigram.

In a final set of experiments, some of the previously investigated models were interpolated linearly with the backoff trigram. The results are shown in Table 4. One can see that the interpolation with the backoff trigram leads to an improvement of up to 9% over the backoff trigram by itself.

Context	Clusters (S,G)	PP (tree)
w_{i-1}	2000,2000	107
w_{i-2}, w_{i-1}	2000,2000	102
$t_{i-4}, t_{i-3}, t_{i-2}, t_{i-1}$	2000,2000	104
g_{i-2}, g_{i-1}	2000,2000	104

Table 4: Interpolation with the backoff trigram

5. Conclusion

An existing clustering algorithm was extended to deal with a hierarchical definition of contexts. This lead to a significant perplexity improvement of up to 12%. The resulting algorithm was used to experiment with different ways of defining the contexts. Although none of the models outperform a backoff trigram, they lead to a perplexity improvement of up to 9% when interpolated with a trigram.

6. REFERENCES

1. E. Brill. Some advances in transformation-based part of speech tagging. In *Twelth National Conference on Artificial Intelligence*, 1994.
2. M. Generet, H. Ney, and F. Wessel. Extensions of absolute discounting for language modeling. In *European Conference on Speech Communication and Technology*, pages vol.2, pp.1245–1248. Madrid, Spain, 1995.
3. S. Martin, J. Liermann, and H. Ney. Algorithms for bigram and trigram word clustering. In *European Conference on Speech Communication and Technology*, pages vol.2, pp.1253–1256. Madrid, Spain, 1995.
4. Hermann Ney, Ute Essen, and Reinhard Kneser. On structuring probabilistic dependencies in stochastic language modelling. *Computer, Speech and Language*, 7:101–138, 1993.
5. R. Rosenfeld. *Adaptive Statistical Language Modeling: A Maximum Entropy Approach*. PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, 1994.
6. Joerg P. Ueberla. More efficient clustering of n-grams for statistical language modeling. In *European Conference on Speech Communication and Technology*, pages pp.1257–1260, vol.2. Madrid, Spain, 1995.