On Efficient Training of Word Classes and Their Application to Recurrent Neural Network Language Models

Rami Botros1, Kazuki Irie1, Martin Sundermeyer1, Hermann Ney1,2

1Human Language Technology and Pattern Recognition, Computer Science Department, RWTH Aachen University, Aachen, Germany
2Spoken Language Processing Group, LIMSI CNRS, Paris, France

botros@rwth-aachen.de, {irie, sundermeyer, ney}@cs.rwth-aachen.de

Abstract

In this paper, we investigated various word clustering methods, by studying two clustering algorithms: Brown clustering and exchange algorithm, and three objective functions derived from different class-based language models (CBLM): two-sided, predictive and conditional models. In particular, we focused on the implementation of the exchange algorithm with improved speed. In total, we compared six clustering methods in terms of runtime and perplexity (PP) of the CBLM on a French corpus, and show that our accelerated implementation of exchange algorithm is up to 114 times faster than the original and around 6 times faster than the best implementation of Brown clustering we could find, while performing about the same (slightly better) in PP. In addition, we conducted a keyword search experiment on the Babel Lithuanian task (IARPA-babel304b-v1.0b), which showed that CBLM improves the word error rate (WER) but not in PP. In addition, we conducted a keyword search experiment on the Babel Lithuanian task (IARPA-babel304b-v1.0b), which showed that CBLM improves the word error rate (WER) but not in PP. Furthermore, we used these clustering techniques for the output layer of a recurrent neural network (RNN) language model (LM) and we show that in terms of PP of the RNN LM, word classes trained under the predictive model perform slightly better than those trained under other criteria we considered.

Index Terms: word clustering, language modeling, neural network based language model, recurrent neural network, long short-term memory

1. Introduction

Word classes were introduced to language modeling to overcome the data sparsity problem [1, 2]. Nowadays, they are used for various tasks in natural language processing including speech recognition [3, 4] and machine translation [5, 6]. For language modeling in particular, they have a new role in accelerating the training of the computationally complex LMs, such as neural network (NN) based LM [7] and maximum entropy LM [8]. Previously, many techniques have been investigated for the unsupervised word clustering, but only few comparisons of these techniques have been reported [9, 4]. The techniques differ mainly in two aspects: the clustering algorithm and the objective function. In this paper, we investigate two clustering algorithms and three objective functions, while considering only hard clustering (each word is assigned to exactly one class). In all techniques we consider, the objective function that the clustering algorithm optimizes is the log-likelihood (equivalently PP) of the CBLM.

\[ F_{g,s} = \sum_{(v,w) \in V^2} N(v,w) \cdot \log p_{g,s}(w|v) \] (1)

where \( V \) is the set of vocabulary, \( N(v,w) \) is the count of word bigram \((v,w)\) in the training text, and we consider three models for the CBLM \( p_{g,s}(w|v) \), which are special cases of:

\[ p_{g,s}(w|v) = p_0(w|g(w)) \cdot p_1(g(w)|s(v)) \], (2)

where \( g \) and \( s \) are class mappings [10]. If \( g \) and \( s \) are the same, we get the two-sided CBLM [1, 2]:

\[ p_{g,s}(w|v) = p_0(w|g(w)) \cdot p_1(g(w)|g(v)) \] (3)

The predictive CBLM is obtained when \( s \) is bijective [6]:

\[ p_{g,s}(u|v) = p_0(w|g(w)) \cdot p_1(g(w)|v) \] (4)

And the conditional CBLM is the one where \( g \) is bijective [4]:

\[ p_{g,s}(w|v) = p_1(w|s(v)) \] (5)

The last two cases are both called one-sided models. In addition, the clustering is called bigram clustering with one word \( v \) in the context as in (1), and trigram clustering with two word context. \( p_0 \) is called class-membership probability and \( p_1 \) class-transition probability.

The rest of the paper is organized as follows: in Section 2, different clustering algorithms and acceleration techniques are reviewed. In Section 3, we experimentally compare word clustering techniques in terms of clustering runtime and their PP performance. In addition, we run a keyword search experiment using CBLM. Finally, in Section 4, we focus on the use of word classes in the NN LM: we compare them in terms of PP of the RNN LM.

2. Clustering Algorithms

2.1. Brown’s Agglomerative Algorithm

We refer to [2] for the complete description of the original algorithm. Here we review the optimized version of Brown’s agglomerative algorithm: Given a fixed number of classes \( G \) to be created, the \( G \) most frequent words are initially put in their own classes. At each step, the next most frequent word is assigned alone to a new class. According to the PP criterion, two from the current \( G+1 \) classes are joined. The algorithm runs in \( O(VG^2) \), where \( V \) is the vocabulary size. For our experiments, we used a highly optimized implementation of this algorithm implemented in [11], as the SRILM toolkit [12] had very high runtime.
2.2. Exchange Algorithm

The exchange algorithm was first introduced in [1]. It starts with an initial mapping and for \( J \) iterations, every word is moved from its class to the class which would result in the best improvement in PP.

For two-sided bigram clustering, its complexity is of order \( O(G^2 \cdot V \cdot I) \) [3]. In the case of conditional and predictive models, the complexity is only of order \( O(G \cdot V \cdot I) \) [4]. In [3], some acceleration techniques for the two-sided case are proposed and the runtime becomes less than quadratic in the number of classes. These techniques are reviewed in the next section.

2.3. Acceleration Methods

Both clustering algorithms presented above can use two common ideas to achieve improvements in runtime:

1. When two candidate classes are examined for agglomeration (Sec. 2.1) or exchange (Sec. 2.2), the resulting PP is not computed from scratch. Instead, it is easier to compute the resulting difference in PP, because the latter only depends on the two classes in question.

2. The calculations can be further accelerated by caching counts that appear in the formula of the difference. These counts have to be updated every time two classes are finally agglomerated or exchanged.

[2, 11] give complete details on how this approach is applied to the Brown clustering. In this section, we focus on the exchange algorithm for the two-sided bigram CBLM (Eq. 3). The corresponding objective function can be expressed in terms of counts as:

\[
F_g = \sum_{w \in V} N(w) \cdot \log N(w) - 2 \cdot \sum_{g_x \in C} N(g_x) \cdot \log N(g_x) + \sum_{(g_x, g_y) \in C^2} N(g_x, g_y) \cdot \log N(g_x, g_y)
\]

(6)

where \( C \) is the set of classes. An overview of the accelerated exchange algorithm is given in Figure 1. In addition to all counts in Eq. (6), the so-called word-class counts, \( N(w, c) \) and \( N(c, w) \) for every word \( w \) and every class \( c \) are cached because they are the main variation terms in the update of class bigram counts. For each word, these counts have to be first generated by iterating through its seen predecessor and the successor classes. Then, when removing a word \( w \) from its class or moving it to a new class (Figure 1), the cached word-class counts, as well as affected class bigram counts need to be updated. The complete count update formulae can be found in [3]. The resulting variation of objective function when moving word \( w \) to class \( k \) (in the inner loop of Figure 1) is as follows:

\[
\Delta F_{\text{move } w \text{ to } k} = \sum_{(g_x, g_y) \in S} N_{\text{new}}(g_x, g_y) \cdot \log N_{\text{new}}(g_x, g_y) - \sum_{(g_x, g_y) \in S} N_{\text{old}}(g_x, g_y) \cdot \log N_{\text{old}}(g_x, g_y) - 2 \cdot (N_{\text{new}}(k) \cdot \log N_{\text{new}}(k)) - 2 \cdot (-N_{\text{old}}(k) \cdot \log N_{\text{old}}(k))
\]

(7)

where \( N_{\text{old}} \) and \( N_{\text{new}} \) are counts before and after the move and \( S \) is the set of class bigrams that have \( k \) as an element. The computation is analogous for removing process. This leaves the two summations over classes in Eq. (6) with considerably less terms, than the summations over all classes. This tentative evaluation is done without updating the actual class uni- and bigram counts stored in the memory; these will only be updated at the final step, when the word is conclusively moved to the best class. This is not only beneficial in terms of efficiency, it also makes it directly feasible to parallelize the read-only inner loop with respect to all classes.

Since Eq. (7) consists of numerous terms of the form \( \log n \cdot \log n \) with \( n \) a natural number, further speed improvements can be gained if all values of such form are precalculated and stored for every required \( n \).

Similar concepts lead to an acceleration of the two-sided trigram clustering algorithm, which we have also implemented.

2.4. Improved Optimization

So far, we have only described algorithms that are greedy. Greedy algorithms only take the locally optimal choice at each step, therefore they can get stuck at bad local minima. This can be improved by running a number of iterations of the relaxed optimization algorithm before the greedy one. [13] provides a comparison between different relaxed optimizations for word clustering and shows that the threshold acceptance (TA) technique works best. In the TA phase, words are only attempted to be moved to the best class different from their current one. Such exchanges can potentially degrade the PP, because in some cases the only possible move without loss is to leave the word in its current class. If the decline in PP is within a given threshold, the move is accepted, otherwise the word is not exchanged. That threshold is decreased for every iteration until the end of TA phase.

[13] gives a method to calculate the initial threshold and shows that the TA phase requires about double the number of iterations as the subsequent greedy phase. We adopt these heuristics.

3. Experimental Results

3.1. Runtime and PP comparison

We conducted the comparative experiments on the Quaero French Corpus1; its statistics are summarized in Table 1. We compared the runtime and PP of implementations for Brown clustering (Liang[11]), unaccelerated exchange algo-

---

1Quaero Programme: http://www.quaero.org
algorithm with TA. The baseline KN5 has test PP of 148.7.

Table 2: Training PP of CBLM, test PP of interpolated models and the clustering runtime for the different bigram clustering techniques on Quaero French. 10 iterations were run for the greedy algorithm and 20 iterations were run in addition for the TA phase for the algorithm with TA. The baseline KN5 has test PP of 148.7.

<table>
<thead>
<tr>
<th>Model</th>
<th>Two-sided</th>
<th>Predictive</th>
<th>Conditional</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td># Classes</td>
<td>Liang</td>
<td>TA</td>
</tr>
<tr>
<td>Train PP</td>
<td>200</td>
<td>417.3</td>
<td>407.9</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>374.4</td>
<td>362.5</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>317.2</td>
<td>313.0</td>
</tr>
<tr>
<td>Test PP</td>
<td>200</td>
<td>139.4</td>
<td>138.4</td>
</tr>
<tr>
<td></td>
<td>400</td>
<td>137.0</td>
<td>136.3</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>136.1</td>
<td>135.5</td>
</tr>
<tr>
<td>Runtime</td>
<td>200</td>
<td>0:23</td>
<td>9:24</td>
</tr>
<tr>
<td>(hour:min)</td>
<td>400</td>
<td>1:34</td>
<td>26:45</td>
</tr>
<tr>
<td></td>
<td>600</td>
<td>4:01</td>
<td>42:59</td>
</tr>
<tr>
<td></td>
<td>800</td>
<td>7:30</td>
<td>68:01</td>
</tr>
<tr>
<td></td>
<td>1000</td>
<td>11:33</td>
<td>73:25</td>
</tr>
</tbody>
</table>

Table 2: Training PP of CBLM, test PP of interpolated models and the clustering runtime for the different bigram clustering techniques on Quaero French. 10 iterations were run for the greedy algorithm and 20 iterations were run in addition for the TA phase for the algorithm with TA. The baseline KN5 has test PP of 148.7.

3.2. Smoothing of the class transition probability

To smooth the CBLMs, we can simply interpolate them with KN5. We further investigated whether we get improvements in smoothing (mkcls) [13] and our own accelerated exchange algorithm on the two-sided bigram clustering. Our software was tested with and without TA. We also used our software to compare the three variations of CBLMs, trained under the bigram criterion. After the class mappings were obtained via the different algorithms, we used the SRILM/FLM toolkits [12] to generate the 4-gram CBLMs (although our class mapping algorithms only optimize with respect to the bigram transition probabilities, we used the resulting word classes to create CBLMs with higher order transition probabilities) and to obtain the test PP. For the baseline LM, we trained a 5-gram count LM with Kneser-Ney smoothing (KN5) [14]. Results are shown in Table 2. The comparison between the Brown algorithm and the exchange algorithm shows that while the difference in test PP is small, the latter resulted in better PP in all our experiments. Among the three variations of the CBLM, we found that the two-sided model performed best in terms of test PP. Overall, the exchange algorithm with two-sided CBLM resulted in the best test PP of 135.5, which is a 9% improvement over the baseline KN5’s test PP of 148.7. We also notice that the TA technique had almost no effect on test PP for the two-sided model, while slightly improving one-sided models (by about 1%, numbers not reported here). For a large number of classes, our implementation was up to 114 times faster than mkcls[13] and around 6 times faster than [11]’s Brown implementation. Precomputing the aforementioned \( n \cdot \log n \) values has sped up the computations by up to 38%. Moreover, the runtimes in Table 2 were obtained while running a single thread: doubling the number of threads reduces the computation time by 47%.

3.3. Bigram vs. Trigram clustering

In addition, we used our exchange algorithm implementation to compare the bigram and trigram criteria for the two-sided model. The results are shown in Figure 2. We found that although the trigram criterion outperforms the bigram criterion for small number of classes, it becomes worse for more classes and the overall best results are obtained with bigram clustering. This is a novel result which updates the conclusion of [3]. In fact, similar comparative experiments have been conducted in [3] for a comparable data size, but the trigram clustering has been tested only with less than 200 classes, which had lead to the conclusion that trigram clustering was superior to the bigram clustering. We see this as a limited conclusion which is only valid for a small number of classes.

3.4. Babel Lithuanian keyword search

We have also conducted an experiment for the Babel keyword search (KWS) task in Lithuanian (IARPA-babel304b-v1.0b) [15]. The statistics of the corpus are presented in Table 1. In this experiment, the \( v_{tlp} \) tune data is used as a validation data and dev data is used as an evaluation data. For the KWS, the
In the NN LM, computations in the output layer are the most expensive. [7] has suggested structuring the output of the NN LM in form of a class-based model, the same technique used by [8] for the maximum entropy LM. [18] has used this technique for RNN LM. The output layer is then decomposed into a product of two terms:

\[ p(w_n | w_{n-1}) = p(w_n | g(w_n), w_{n-1})p(g(w_n) | w_{n-1}^{-1}) \]  

\[ (8) \]

[18] has used frequency-based classes while [19] has shown that two-sided bigram classes perform better. We tested one-sided classes as well as two-sided trigram classes on the Quaero...
7. References


