



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

Rami Botros¹, Kazuki Irie¹, Martin Sundermeyer¹, Hermann Ney^{1,2}

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 the keyword search performance. 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,

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

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:

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

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

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

The *predictive* CBLM is obtained when s is bijective [6]:

$$p_{g,s}(w|v) = p_0(w|g(w)) \cdot p_1(g(w)|v) \quad (4)$$

And the *conditional* CBLM is the one where g is bijective [4]:

$$p_{g,s}(w|v) = p_1(w|s(v)) \quad (5)$$

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 I 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:

$$F_g = \sum_{w \in V} N(w) \cdot \log N(w) - 2 \cdot \sum_{g_x \in C} N(g_x) \cdot \log N(g_x) + \sum_{(g_x, g_y) \in C^2} N(g_x, g_y) \cdot \log N(g_x, g_y) \quad (6)$$

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:

$$\begin{aligned} \Delta F_{\text{move } w \text{ to } k} = & \sum_{(g_x, g_y) \in S} N_{\text{new}}(g_x, g_y) \cdot \log N_{\text{new}}(g_x, g_y) \\ & - \sum_{(g_x, g_y) \in S} N_{\text{old}}(g_x, g_y) \cdot \log N_{\text{old}}(g_x, g_y) \\ & - 2 \cdot (N_{\text{new}}(k) \cdot \log N_{\text{new}}(k)) \\ & - 2 \cdot (-N_{\text{old}}(k) \cdot \log N_{\text{old}}(k)) \end{aligned} \quad (7)$$

where N_{old} and N_{new} are counts before and after the *move* and S is the set of class bigrams that have k as an element. The

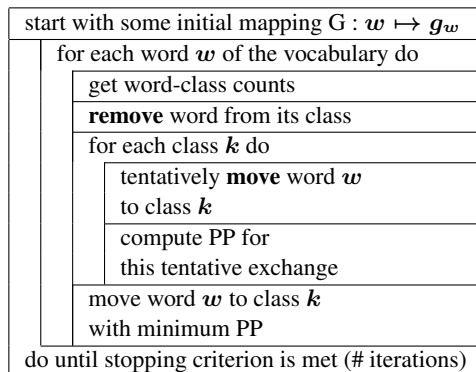


Figure 1: Accelerated exchange algorithm [3]

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 $n \cdot \log n$, 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>

Model	# Classes	Two-sided				Predictive	Conditional
		Liang	mkcls TA	TA	Greedy	TA	TA
Train PP	200	417.3	407.9	407.8	414.9	408.4	283.8
	400	374.4	362.5	362.6	366.7	362.3	256.0
	1000	317.2	313.0	312.9	317.1	299.4	226.4
Test PP	200	139.4	138.4	138.6	138.4	141.4	141.1
	400	137.0	136.3	136.2	136.6	141.8	141.6
	1000	136.1	135.5	135.8	135.9	142.5	143.7
Runtime (hour:min)	200	0:23	9:24	0:16	0:05	0:16	0:17
	400	1:34	26:45	0:48	0:16	0:32	0:34
	600	4:01	42:59	2:03	0:41	0:49	0:53
	800	7:30	68:01	3:15	1:05	1:04	1:10
	1000	11:33	73:25	5:30	1:50	1:20	1:26

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.

Corpus		# Words	VOC
Quaero French	train	27 M	179 K
	dev	35 K	
	test	41 K	
Babel Lithuanian	train	303 K	5.5 K
	vllp_tune	24 K	
	dev	76 K	

Table 1: Corpus Statistics

rithm (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%.

3.2. 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

	# Classes	Not Smoothed	Smoothed
Two-sided	200	138.8	138.4
	600	136.2	135.8
	1000	136.4	135.9
Predictive	200	143.7	141.4
	600	145.0	142.2
	1000	145.0	142.5
Conditional	200	145.9	141.1
	600	146.9	142.5
	1000	147.2	143.7

Table 3: PP on Quaero test data of the interpolation model between word KN5 and the class based model with and without smoothing of the transition probability.

PP by smoothing the transition probability of the CBLM. The results are presented in Table 3. It shows that it has a marginal effect on the two-sided model, while improving the one-sided models by up to 3% relative in test PP. All PPs in Table 2 are computed with the CBLM with smoothed transition probabilities.

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 led 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 vllp_tune data is used as a validation data and dev data is used as an evaluation data. For the KWS, the

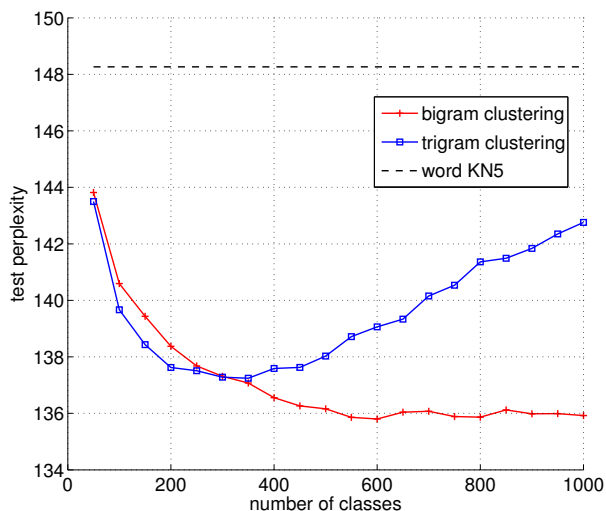


Figure 2: Test PP of the two-sided CBLM trained with bigram clustering and trigram clustering, after interpolation with KN5.

LM	vllp_tune		dev	
	PP	PP	WER [%]	MTWV
KN2	162.8	153.8	61.7	0.3835
+ CBLM	160.9	151.8	61.0	0.3829

Table 4: KWS experiments on Lithuanian VLLP

bigram count LM has been reported to perform best [16]; our baseline is therefore a bigram LM (KN2). We trained a two-sided bigram class-based LM with bigram clustering. We used vllp_tune set to optimize the number of classes: we found 100 to be optimal in the range between 100 and 2000. In order to use the CBLM directly during recognition, it has been expanded to the word level ARPA format LM using the SRILM toolkit [12].

The acoustic model used for lattice generation is a Gaussian mixture model trained on multilingual features and includes speaker adapted training and minimum phone error. We refer to [15] for the details of the system. The KWS is evaluated by the maximum term weighted value (MTWV) [17]: higher MTWV is better. The experimental results are presented in Table 4. We found that despite some improvement in PP and in WER, CBLM did not improve the KWS.

4. Word clustering for the RNN LM

In this section, we focus on the recent interest for using word classes to speed up the training of NN LM.

4.1. Output Layer Factorization for NN LM

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:

$$p(w_n|w_1^{n-1}) = p(w_n|g(w_n), w_1^{n-1})p(g(w_n)|w_1^{n-1}) \quad (8)$$

[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

# Classes	Freq.	Bigram			Trigram
	-	Two-sided	Pred.	Cond.	Two-sided
250	135.6	116.1	117.6	120.4	117.8
400	133.0	117.0	116.4	120.2	126.2
700	133.9	116.2	115.2	121.2	118.3
2000	135.7	115.4	115.1	123.6	125.3

Table 5: Test PP of the LSTM-RNNLM using different clustering techniques

French (Table 1). We trained the long short-term memory recurrent neural network LM (LSTM-RNN LM) [20] with an input layer of size 200 and a single hidden layer of the same size, using [21]. Results are shown in Table 5. While PP-based classes outperform the frequency-based ones, trigram classes perform worse than bigram classes. Predictive classes offer a viable alternative to two-sided ones and they are faster to generate.

5. Conclusion

We have implemented an improved exchange algorithm and compared it with the Brown clustering. Our implementation was multiple times faster with comparable PP performance. We investigated the effect of relaxed optimization and found that improvements for training PPs was around 1% relative, which did not carry over to the test PP. The speedup enabled us to perform experiments with the trigram clustering for a high number of classes and we observe that it was not beneficial over the bigram clustering: this is a novel result which extends the earlier work which had concluded that the trigram clustering outperforms the bigram clustering.

Moreover, two one-sided CBLMs were compared to the two-sided one. Both resulted in worse test PP, but their computational complexity is only linear with the number of classes, which could be beneficial when we consider a high number of classes. However, the two-sided clustering without TA could be an acceptable compromise, as it results in better PP, while maintaining a competitive runtime with the one-sided models.

In addition, we used the CBLM for a keyword search experiment: while it improved the PP and WER over the word LM, this did not transfer to MTWV improvement.

For the application to the RNN LM, the two-sided trigram clustering was, again, not beneficial. Among the bigram clustering techniques, the predictive model gave the best results. For future experiments, it might be beneficial to try trigram clustering with a cross-validation to avoid the overfit.

6. Acknowledgements

We thank Pavel Golik for helping us with the Babel keyword search experiments. H. Ney was partially supported by a senior chair award from DIGITEO, a French research cluster in Île-de-France. Supported by the Intelligence Advanced Research Projects Activity (IARPA) via Department of Defense U.S. Army Research Laboratory (DoD/ARL) contract no. W911NF-12-C-0012. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright annotation thereon. Disclaimer: The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of IARPA, DoD/ARL, or the U.S. Government.

7. References

- [1] R. Kneser and H. Ney, "Forming word classes by statistical clustering for statistical language modelling," in *Proc. of the First International Conference on Quantitative Linguistics (QUALICO)*, Trier, Germany, 1991, pp. 221–226.
- [2] P. F. Brown, P. V. Desouza, R. L. Mercer, V. J. D. Pietra, and J. C. Lai, "Class-based n-gram models of natural language," *Computational linguistics*, vol. 18, no. 4, pp. 467–479, 1992.
- [3] S. Martin, J. Liermann, and H. Ney, "Algorithms for bigram and trigram word clustering," *Speech communication*, vol. 24, no. 1, pp. 19–37, 1998.
- [4] E. W. Whittaker and P. C. Woodland, "Efficient class-based language modelling for very large vocabularies," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Salt Lake City, UT, USA, May 2001, pp. 545–548.
- [5] J. Wuebker, S. Peitz, F. Rietig, and H. Ney, "Improving statistical machine translation with word class models," in *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Seattle, WA, USA, Oct. 2013, pp. 1377–1381.
- [6] J. Uszkoreit and T. Brants, "Distributed word clustering for large scale class-based language modeling in machine translation," in *Proc. of the Annual Meeting of the Association for Computational Linguistics (ACL)*, Columbus, OH, USA, 2008, pp. 755–762.
- [7] F. Morin and Y. Bengio, "Hierarchical probabilistic neural network language model," in *Proc. of the international workshop on artificial intelligence and statistics (AISTATS)*, vol. 5, Barbados, Jan. 2005, pp. 246–252.
- [8] J. Goodman, "Classes for fast maximum entropy training," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, vol. 1, Salt Lake City, UT, USA, May 2001, pp. 561–564.
- [9] J. Gao, J. Goodman, J. Miao *et al.*, "The use of clustering techniques for language modeling—application to asian languages," *Computational Linguistics and Chinese Language Processing*, vol. 6, no. 1, pp. 27–60, 2001.
- [10] H. Ney, U. Essen, and R. Kneser, "On structuring probabilistic dependences in stochastic language modelling," *Computer Speech & Language*, vol. 8, no. 1, pp. 1–38, 1994.
- [11] P. Liang, "Semi-supervised learning for natural language," Master's thesis, Massachusetts Institute of Technology, 2005.
- [12] A. Stolcke *et al.*, "SRILM—an extensible language modeling toolkit," in *Proc. Interspeech*, Denver, CO, USA, 2002.
- [13] F. J. Och, "Maximum-likelihood-schätzung von wortkategorien mit verfahren der kombinatorischen optimierung," *Studienarbeit, Friedrich-Alexander-Universität, Erlangen-Nürnberg, Germany*, 1995.
- [14] M. Sundermeyer, R. Schlüter, and H. Ney, "On the estimation of discount parameters for language model smoothing," in *Proc. Interspeech*, Florence, Italy, Aug. 2011, pp. 1433–1436.
- [15] P. Golik, Z. Tüske, R. Schlüter, and H. Ney, "Multilingual features based keyword search for very low-resource languages," in *Proc. Interspeech*, Dresden, Germany, Sep. 2015.
- [16] K. Knill, M. J. Gales, S. P. Rath, P. C. Woodland, C. Zhang, and S.-X. Zhang, "Investigation of multilingual deep neural networks for spoken term detection," Olomouc, Czech Republic, Dec. 2013, pp. 138–143.
- [17] J. G. Fiscus, J. Ajot, J. S. Garofolo, and G. Doddington, "Results of the 2006 spoken term detection evaluation," in *Proc. Special Interest Group on Information Retrieval (SIGIR)*, vol. 7, Amsterdam, Netherlands, 2007, pp. 51–57.
- [18] T. Mikolov, S. Kombrink, L. Burget, J. H. Cernocky, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Prague, Czech Republic, May 2011, pp. 5528–5531.
- [19] G. Zweig and K. Makarychev, "Speed regularization and optimality in word classing," in *Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing (ICASSP)*, Vancouver, Canada, May 2013, pp. 8237–8241.
- [20] M. Sundermeyer, R. Schlüter, and H. Ney, "LSTM neural networks for language modeling," in *Proc. Interspeech*, Portland, OR, USA, Sep. 2012, pp. 194–197.
- [21] —, "rwthlm the RWTH Aachen University neural network language modeling toolkit," in *Proc. Interspeech*, Singapore, Sep. 2014, pp. 2093–2097.