Distributed Word Clustering for Large Scale Class-Based Language Modeling in Machine Translation

Jakob Uszkoreit* **Thorsten Brants**

Google, Inc.
1600 Amphitheatre Parkway
Mountain View, CA 94303, USA
{uszkoreit,brants}@google.com

Abstract

In statistical language modeling, one technique to reduce the problematic effects of data sparsity is to partition the vocabulary into equivalence classes. In this paper we investigate the effects of applying such a technique to higher-order n-gram models trained on large corpora. We introduce a modification of the exchange clustering algorithm with improved efficiency for certain partially class-based models and a distributed version of this algorithm to efficiently obtain automatic word classifications for large vocabularies (>1 million words) using such large training corpora (>30 billion tokens). The resulting clusterings are then used in training partially class-based language models. We show that combining them with word-based n-gram models in the log-linear model of a state-of-the-art statistical machine translation system leads to improvements in translation quality as indicated by the BLEU score.

1 Introduction

A statistical language model assigns a probability $P(w)$ to any given string of words $w_1^m = w_1, ..., w_m$. In the case of n-gram language models this is done by factoring the probability:

$$P(w_1^m) = \prod_{i=1}^{m} P(w_i|w_1^{i-1})$$

and making a Markov assumption by approximating this by:

$$\prod_{i=1}^{m} P(w_i|w_1^{i-1}) \approx \prod_{i=1}^{m} p(w_i|w_{i-n+1}^{i-1})$$

Even after making the Markov assumption and thus treating all strings of preceding words as equal which do not differ in the last $n-1$ words, one problem n-gram language models suffer from is that the training data is too sparse to reliably estimate all conditional probabilities $P(w_i|w_1^{i-1})$.

Class-based n-gram models are intended to help overcome this data sparsity problem by grouping words into equivalence classes rather than treating them as distinct words and thus reducing the number of parameters of the model (Brown et al., 1990). They have often been shown to improve the performance of speech recognition systems when combined with word-based language models (Martin et al., 1998; Whittaker and Woodland, 2001). However, in the area of statistical machine translation, especially in the context of large training corpora, fewer experiments with class-based n-gram models have been performed with mixed success (Raab, 2006).

Class-based n-gram models have also been shown to benefit from their reduced number of parameters when scaling to higher-order n-grams (Goodman and Gao, 2000), and even despite the increasing size and decreasing sparsity of language model training corpora (Brants et al., 2007), class-based n-gram models might lead to improvements when increasing the n-gram order.

When training class-based n-gram models on large corpora and large vocabularies, one of the problems arising is the scalability of the typical clustering algorithms used for obtaining the word classification. Most often, variants of the *exchange algorithm* (Kneser and Ney, 1993; Martin et al., 1998) or the agglomerative clustering algorithm presented in (Brown et al., 1990) are used, both of which have prohibitive runtimes when clustering large vocabularies on the basis of large training corpora with a sufficiently high number of classes.

In this paper we introduce a modification of the exchange algorithm with improved efficiency and then present a distributed version of the modified algorithm, which makes it feasible to obtain word clas-

* Parts of this research were conducted while the author studied at the Berlin Institute of Technology

Proceedings of ACL-08: HLT, pages 755–762,
Columbus, Ohio, USA, June 2008.

sifications using billions of tokens of training data. We then show that using partially class-based language models trained using the resulting classifications together with word-based language models in a state-of-the-art statistical machine translation system yields improvements despite the very large size of the word-based models used.

2 Class-Based Language Modeling

By partitioning all N_v words of the vocabulary into N_c sets, with $c(w)$ mapping a word onto its equivalence class and $c(w_i^j)$ mapping a sequence of words onto the sequence of their respective equivalence classes, a typical class-based n-gram model approximates $P(w_i|w_1^{i-1})$ with the two following component probabilities:

$$P(w_i|w_1^{i-1}) \approx p_0(w_i|c(w_i)) \cdot p_1(c(w_i)|c(w_{i-n+1}^{i-1})) \quad (1)$$

thus greatly reducing the number of parameters in the model, since usually N_c is much smaller than N_v.

In the following, we will call this type of model a *two-sided class-based model*, as both the history of each n-gram, the sequence of words conditioned on, as well as the predicted word are replaced by their class.

Once a partition of the words in the vocabulary is obtained, two-sided class-based models can be built just like word-based n-gram models using existing infrastructure. In addition, the size of the model is usually greatly reduced.

2.1 One-Sided Class-Based Models

Two-sided class-based models received most attention in the literature. However, several different types of mixed word and class models have been proposed for the purpose of improving the performance of the model (Goodman, 2000), reducing its size (Goodman and Gao, 2000) as well as lowering the complexity of related clustering algorithms (Whittaker and Woodland, 2001).

In (Emami and Jelinek, 2005) a clustering algorithm is introduced which outputs a separate clustering for each word position in a trigram model. In the experimental evaluation, the authors observe the largest improvements using a specific clustering for the last word of each trigram but no clustering at all for the first two word positions. Generalizing this leads to arbitrary order class-based n-gram models of the form:

$$P(w_i|w_1^{i-1}) \approx p_0(w_i|c(w_i)) \cdot p_1(c(w_i)|w_{i-n+1}^{i-1}) \quad (2)$$

which we will call *predictive class-based models* in the following sections.

3 Exchange Clustering

One of the frequently used algorithms for automatically obtaining partitions of the vocabulary is the exchange algorithm (Kneser and Ney, 1993; Martin et al., 1998). Beginning with an initial clustering, the algorithm greedily maximizes the log likelihood of a two-sided class bigram or trigram model as described in Eq. (1) on the training data. Let V be the set of words in the vocabulary and C the set of classes. This then leads to the following optimization criterion, where $N(w)$ and $N(c)$ denote the number of occurrences of a word w or a class c in the training data and $N(c,d)$ denotes the number of occurrences of some word in class c followed by a word in class d in the training data:

$$\begin{aligned} \hat{C} \quad = \quad & \operatorname*{argmax}_C \sum_{w \in V} N(w) \cdot \log N(w) + \\ & + \sum_{c \in C, d \in C} N(c,d) \cdot \log N(c,d) - \\ & -2 \cdot \sum_{c \in C} N(c) \cdot \log N(c) \end{aligned} \quad (3)$$

The algorithm iterates over all words in the vocabulary and tentatively moves each word to each cluster. The change in the optimization criterion is computed for each of these tentative moves and the exchange leading to the highest increase in the optimization criterion (3) is performed. This procedure is then repeated until the algorithm reaches a local optimum.

To be able to efficiently calculate the changes in the optimization criterion when exchanging a word, the counts in Eq. (3) are computed once for the initial clustering, stored, and afterwards updated when a word is exchanged.

Often only a limited number of iterations are performed, as letting the algorithm terminate in a local optimum can be computationally impractical.

3.1 Complexity

The implementation described in (Martin et al., 1998) uses a memory saving technique introducing a binary search into the complexity estimation. For the sake of simplicity, we omit this detail in the following complexity analysis. We also do not employ this optimization in our implementation.

The worst case complexity of the exchange algorithm is quadratic in the number of classes. However,

Input: The fixed number of clusters N_c
Compute initial clustering
while *clustering changed in last iteration* **do**
  forall $w \in V$ **do**
    forall $c \in C$ **do**
      move word w tentatively to cluster c
      compute updated optimization criterion
    move word w to cluster maximizing optimization criterion

Algorithm 1: Exchange Algorithm Outline

the average case complexity can be reduced by updating only the counts which are actually affected by moving a word from one cluster to another. This can be done by considering only those sets of clusters for which $N(w,c) > 0$ or $N(c,w) > 0$ for a word w about to be exchanged, both of which can be calculated efficiently when exchanging a word. The algorithm scales linearly in the size of the vocabulary.

With N_c^{pre} and N_c^{suc} denoting the average number of clusters preceding and succeeding another cluster, B denoting the number of distinct bigrams in the training corpus, and I denoting the number of iterations, the worst case complexity of the algorithm is in:

$$\mathcal{O}(I \cdot (2 \cdot B + N_v \cdot N_c \cdot (N_c^{pre} + N_c^{suc})))$$

When using large corpora with large numbers of bigrams the number of required updates can increase towards the quadratic upper bound as N_c^{pre} and N_c^{suc} approach N_c. For a more detailed description and further analysis of the complexity, the reader is referred to (Martin et al., 1998).

4 Predictive Exchange Clustering

Modifying the exchange algorithm in order to optimize the log likelihood of a predictive class bigram model, leads to substantial performance improvements, similar to those previously reported for another type of one-sided class model in (Whittaker and Woodland, 2001).

We use a predictive class bigram model as given in Eq. (2), for which the maximum-likelihood probability estimates for the n-grams are given by their relative frequencies:

$$\begin{aligned} P(w_i|w_1^{i-1}) &\approx p_0(w_i|c(w_i)) \cdot p_1(c(w_i)|w_{i-1}) && (4) \\ &= \frac{N(w_i)}{N(c(w_i))} \cdot \frac{N(w_{i-1}, c(w_i))}{N(w_{i-1})} && (5) \end{aligned}$$

where $N(w)$ again denotes the number of occurrences of the word w in the training corpus and $N(v,c)$ the number of occurrences of the word v followed by some word in class c. Then the following optimization criterion can be derived, with $F(C)$ being the log likelihood function of the predictive class bigram model given a clustering C:

$$\begin{aligned} F(C) &= \sum_{w \in V} N(w) \cdot \log p(w|c(w)) \\ &\quad + \sum_{v \in V, c \in C} N(v,c) \cdot \log p(c|v) && (6) \\ &= \sum_{w \in V} N(w) \cdot \log \frac{N(w)}{N(c(w))} \\ &\quad + \sum_{v \in V, c \in C} N(v,c) \cdot \log \frac{N(v,c)}{N(v)} && (7) \\ &= \sum_{w \in V} N(w) \cdot \log N(w) \\ &\quad - \sum_{w \in V} N(w) \cdot \log N(c(w)) \\ &\quad + \sum_{v \in V, c \in C} N(v,c) \cdot \log N(v,c) \\ &\quad - \sum_{v \in V, c \in C} N(v,c) \cdot \log N(v) && (8) \end{aligned}$$

The very last summation of Eq. (8) now effectively sums over all occurrences of all words and thus cancels out with the first summation of (8) which leads to:

$$\begin{aligned} F(C) &= \sum_{v \in V, c \in C} N(v,c) \cdot \log N(v,c) \\ &\quad - \sum_{w \in V} N(w) \cdot \log N(c(w)) && (9) \end{aligned}$$

In the first summation of Eq. (9), for a given word v only the set of classes which contain at least one word w for which $N(v,w) > 0$ must be considered, denoted by $suc(v)$. The second summation is equivalent to $\sum_{c \in C} N(c) \cdot \log N(c)$. Thus the further simplified criterion is:

$$\begin{aligned} F(C) &= \sum_{v \in V, c \in suc(v)} N(v,c) \cdot \log N(v,c) \\ &\quad - \sum_{c \in C} N(c) \cdot \log N(c) && (10) \end{aligned}$$

When exchanging a word w between two classes c and c', only two summands of the second summation of Eq. (10) are affected. The first summation can be updated by iterating over all bigrams ending in the exchanged word. Throughout one iteration of the algorithm, in which for each word in the vocabulary each possible move to another class is evaluated, this

amounts to the number of distinct bigrams in the training corpus B, times the number of clusters N_c. Thus the worst case complexity using the modified optimization criterion is in:

$$\mathcal{O}(I \cdot N_c \cdot (B + N_v))$$

Using this optimization criterion has two effects on the complexity of the algorithm. The first difference is that in contrast to the exchange algorithm using a two sided class-based bigram model in its optimization criterion, only two clusters are affected by moving a word. Thus the algorithm scales linearly in the number of classes. The second difference is that B dominates the term $B + N_v$ for most corpora and scales far less than linearly with the vocabulary size, providing a significant performance advantage over the other optimization criterion, especially when large vocabularies are used (Whittaker and Woodland, 2001).

For efficiency reasons, an exchange of a word between two clusters is separated into a *remove* and a *move* procedure. In each iteration the *remove* procedure only has to be called once for each word, while for a given word *move* is called once for every cluster to compute the consequences of the tentative exchanges. An outline of the *move* procedure is given below. The *remove* procedure is similar.

Input: A word w, and a destination cluster c
Result: The change in the optimization criterion when moving w to cluster c

$delta \leftarrow N(c) \cdot \log N(c)$
$N'(c) \leftarrow N(c) - N(w)$
$delta \leftarrow delta - N'(c) \cdot \log N'(c)$
if *not a tentative move* **then**
  $N(c) \leftarrow N'(c)$
forall $v \in suc(w)$ **do**
  $delta \leftarrow delta - N(v, c) \cdot \log N(v, c)$
  $N'(v, c) \leftarrow N(v, c) - N(v, w)$
  $delta \leftarrow delta + N'(v, c) \cdot \log N'(v, c)$
  if *not a tentative move* **then**
    $N(v, c) \leftarrow N'(v, c)$
return *delta*

Procedure `MoveWord`

5 Distributed Clustering

When training on large corpora, even the modified exchange algorithm would still require several days if not weeks of CPU time for a sufficient number of iterations.

To overcome this we introduce a novel *distributed exchange algorithm*, based on the modified exchange algorithm described in the previous section. The vocabulary is randomly partitioned into sets of roughly equal size. With each word w in one of these sets, all words v preceding w in the corpus are stored with the respective bigram count $N(v, w)$.

The clusterings generated in each iteration as well as the initial clustering are stored as the set of words in each cluster, the total number of occurrences of each cluster in the training corpus, and the list of words preceeding each cluster. For each word w in the predecessor list of a given cluster c, the number of times w occurs in the training corpus before any word in c, $N(w, c)$, is also stored.

Together with the counts stored with the vocabulary partitions, this allows for efficient updating of the terms in Eq. (10).

The initial clustering together with all the required counts is created in an initial iteration by assigning the n-th most frequent word to cluster $n \bmod N_c$. While (Martin et al., 1998) and (Emami and Jelinek, 2005) observe that the initial clustering does not seem to have a noticeable effect on the quality of the resulting clustering or the convergence rate, the intuition behind this method of initialization is that it is unlikely for the most frequent words to be clustered together due to their high numbers of occurrences.

In each subsequent iteration each one of a number of workers is assigned one of the partitions of the words in the vocabulary. After loading the current clustering, it then randomly chooses a subset of these words of a fixed size. For each of the selected words the worker then determines to which cluster the word is to be moved in order to maximize the increase in log likelihood, using the count updating procedures described in the previous section. All changes a worker makes to the clustering are accumulated locally in delta data structures. At the end of the iteration all deltas are merged and applied to the previous clustering, resulting in the complete clustering loaded in the next iteration.

This algorithm fits well into the *MapReduce* programming model (Dean and Ghemawat, 2004) that we used for our implementation.

5.1 Convergence

While the greedy non-distributed exchange algorithm is guaranteed to converge as each exchange increases the log likelihood of the assumed bigram model, this is not necessarily true for the distributed exchange algorithm. This stems from the fact that the change in log likelihood is calculated by each worker under the assumption that no other changes to the clustering are performed by other workers in

this iteration. However, if in each iteration only a rather small and randomly chosen subset of all words are considered for exchange, the intuition is that the remaining words still define the parameters of each cluster well enough for the algorithm to converge.

In (Emami and Jelinek, 2005) the authors observe that only considering a subset of the vocabulary of half the size of the complete vocabulary in each iteration does not affect the time required by the exchange algorithm to converge. Yet each iteration is sped up by approximately a factor of two. The quality of class-based models trained using the resulting clusterings did not differ noticeably from those trained using clusterings for which the full vocabulary was considered in each iteration. Our experiments showed that this also seems to be the case for the distributed exchange algorithm. While considering very large subsets of the vocabulary in each iteration can cause the algorithm to not converge at all, considering only a very small fraction of the words for exchange will increase the number of iterations required to converge. In experiments we empirically determined that choosing a subset of roughly a third of the size of the full vocabulary is a good balance in this trade-off. We did not observe the algorithm to not converge unless we used fractions above half of the vocabulary size.

We typically ran the clustering for 20 to 30 iterations after which the number of words exchanged in each iteration starts to stabilize at less than 5 percent of the vocabulary size. Figure 1 shows the number of words exchanged in each of 34 iterations when clustering the approximately 300,000 word vocabulary of the Arabic side of the English-Arabic parallel training data into 512 and 2,048 clusters.

Despite a steady reduction in the number of words exchanged per iteration, we observed the convergence in regards to log-likelihood to be far from monotone. In our experiments we were able to achieve significantly more monotone and faster convergence by employing the following heuristic. As described in Section 5, we start out the first iteration with a random partition of the vocabulary into subsets each assigned to a specific worker. However, instead of keeping this assignment constant throughout all iterations, after each iteration the vocabulary is partitioned anew so that all words from any given cluster are considered by the same worker in the next iteration. The intuition behind this heuristic is that as the clustering becomes more coherent, the information each worker has about groups of similar words is becoming increasingly accurate. In our experiments this heuristic lead to almost monotone convergence in log-likelihood. It also reduced the number of iterations required to converge by up to a factor of three.

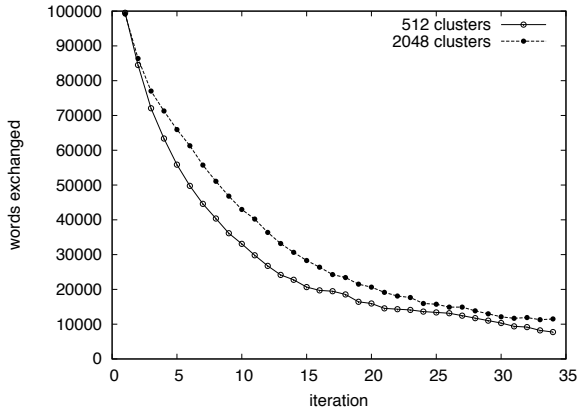

Figure 1: Number of words exchanged per iteration when clustering the vocabulary of the Arabic side of the English-Arabic parallel training data (347 million tokens).

5.2 Resource Requirements

The runtime of the distributed exchange algorithm depends highly on the number of distinct bigrams in the training corpus. When clustering the approximately 1.5 million word vocabulary of a 405 million token English corpus into 1,000 clusters, one iteration takes approximately 5 minutes using 50 workers based on standard hardware running the Linux operating system. When clustering the 0.5 million most frequent words in the vocabulary of an English corpus with 31 billion tokens into 1,000 clusters, one iteration takes approximately 30 minutes on 200 workers.

When scaling up the vocabulary and corpus sizes, the current bottleneck of our implementation is loading the current clustering into memory. While the memory requirements decrease with each iteration, during the first few iterations a worker typically still needs approximately 2 GB of memory to load the clustering generated in the previous iteration when training 1,000 clusters on the 31 billion token corpus.

6 Experiments

We trained a number of predictive class-based language models on different Arabic and English corpora using clusterings trained on the complete data of the same corpus. We use the distributed training and application infrastructure described in (Brants et al., 2007) with modifications to allow the training of predictive class-based models and their application in the decoder of the machine translation system.

For all models used in our experiments, both word- and class-based, the smoothing method used was *Stupid Backoff* (Brants et al., 2007). Models with Stupid Backoff return scores rather than normalized probabilities, thus perplexities cannot be calculated for these models. Instead we report BLEU scores (Papineni et al., 2002) of the machine translation system using different combinations of word- and class-based models for translation tasks from English to Arabic and Arabic to English.

6.1 Training Data

For English we used three different training data sets:
en_target: The English side of Arabic-English and Chinese-English parallel data provided by LDC (405 million tokens).
en_ldcnews: Consists of several English news data sets provided by LDC (5 billion tokens).
en_webnews: Consists of data collected up to December 2005 from web pages containing primarily English news articles (31 billion tokens).
A fourth data set, *en_web*, was used together with the other three data sets to train the large word-based model used in the second machine translation experiment. This set consists of general web data collected in January 2006 (2 trillion tokens).

For Arabic we used the following two different training data sets:
ar_gigaword: Consists of several Arabic news data sets provided by LDC (629 million tokens).
ar_webnews: Consists of data collected up to December 2005 from web pages containing primarily Arabic news articles (approximately 600 million tokens).

6.2 Machine Translation Results

Given a sentence $\mathbf{f}$ in the source language, the *machine translation* problem is to automatically produce a translation $\hat{\mathbf{e}}$ in the target language. In the subsequent experiments, we use a phrase-based statistical machine translation system based on the log-linear formulation of the problem described in (Och and Ney, 2002):

$$\begin{aligned} \hat{\mathbf{e}} &= \operatorname*{argmax}_{\mathbf{e}} p(\mathbf{e}|\mathbf{f}) \\ &= \operatorname*{argmax}_{\mathbf{e}} \sum_{m=1}^{M} \lambda_m h_m(\mathbf{e}, \mathbf{f}) \end{aligned} \quad (11)$$

where $\{h_m(\mathbf{e}, \mathbf{f})\}$ is a set of M *feature functions* and $\{\lambda_m\}$ a set of *weights*. We use each predictive class-based language model as well as a word-based model as separate feature functions in the log-linear combination in Eq. (11). The weights are trained using minimum error rate training (Och, 2003) with BLEU score as the objective function.

The *dev* and *test* data sets contain parts of the 2003, 2004 and 2005 Arabic NIST MT evaluation sets among other parallel data. The blind test data used is the "NIST" part of the 2006 Arabic-English NIST MT evaluation set, and is not included in the training data.

For the first experiment we trained predictive class-based 5-gram models using clusterings with 64, 128, 256 and 512 clusters[1] on the *en_target* data. We then added these models as additional features to the log linear model of the Arabic-English machine translation system. The word-based language model used by the system in these experiments is a 5-gram model also trained on the *en_target* data set.

Table 1 shows the BLEU scores reached by the translation system when combining the different class-based models with the word-based model in comparison to the BLEU scores by a system using only the word-based model on the Arabic-English translation task.

	dev	**test**	**nist06**
word-based only	0.4085	0.3498	0.5088
64 clusters	0.4122	0.3514	0.5114
128 clusters	0.4142	0.3530	0.5109
256 clusters	0.4141	0.3536	0.5076
512 clusters	0.4120	0.3504	0.5140

Table 1: BLEU scores of the Arabic English system using models trained on the English *en_target* data set

Adding the class-based models leads to small improvements in BLEU score, with the highest improvements for both *dev* and *nist06* being statistically significant [2].

In the next experiment we used two predictive class-based models, a 5-gram model with 512 clusters trained on the *en_target* data set and a 6-gram model also using 512 clusters trained on the *en_ldcnews* data set. We used these models in addition to a word-based 6-gram model created by combining models trained on all four English data sets.

Table 2 shows the BLEU scores of the machine translation system using only this word-based model, the scores after adding the class-based model trained on the *en_target* data set and when using all three models.

[1] The *beginning of sentence*, *end of sentence* and *unkown word* tokens were each treated as separate clusters

[2] Differences of more than 0.0051 are statistically significant at the 0.05 level using bootstrap resampling (Noreen, 1989; Koehn, 2004)

	dev	test	nist06
word-based only	0.4677	0.4007	0.5672
with *en_target*	0.4682	0.4022	0.5707
all three models	0.4690	0.4059	0.5748

Table 2: BLEU scores of the Arabic English system using models trained on various data sets

For our experiment with the English Arabic translation task we trained two *5*-gram predictive class-based models with 512 clusters on the Arabic *ar_gigaword* and *ar_webnews* data sets. The word-based Arabic 5-gram model we used was created by combining models trained on the Arabic side of the parallel training data (347 million tokens), the *ar_gigaword* and *ar_webnews* data sets, and additional Arabic web data.

	dev	test	nist06
word-based only	0.2207	0.2174	0.3033
with *ar_webnews*	0.2237	0.2136	0.3045
all three models	0.2257	0.2260	0.3318

Table 3: BLEU scores of the English Arabic system using models trained on various data sets

As shown in Table 3, adding the predictive class-based model trained on the *ar_webnews* data set leads to small improvements in *dev* and *nist06* scores but causes the *test* score to decrease. However, adding the class-based model trained on the *ar_gigaword* data set to the other class-based and the word-based model results in further improvement of the *dev* score, but also in large improvements of the *test* and *nist06* scores.

We performed experiments to eliminate the possibility of data overlap between the training data and the machine translation test data as cause for the large improvements. In addition, our experiments showed that when there is overlap between the training and test data, the class-based models lead to lower scores as long as they are trained only on data also used for training the word-based model. One explanation could be that the domain of the *ar_gigaword* corpus is much closer to the domain of the test data than that of other training data sets used. However, further investigation is required to explain the improvements.

6.3 Clusters

The clusters produced by the distributed algorithm vary in their size and number of occurrences. In a clustering of the *en_target* data set with 1,024 clusters, the cluster sizes follow a typical long-tailed distribution with the smallest cluster contain-

Bai Bi Bu Cai Cao Chang Chen Cheng Chou Chuang Cui Dai Deng Ding Du Duan Fan Fu Gao Ge Geng Gong Gu Guan Han Hou Hsiao Hsieh Hsu Hu Huang Huo Jiang Jiao Juan Kang Kuang Kuo Li Liang Liao Lin Liu Lu Luo Mao Meets Meng Mi Miao Mu Niu Pang Pi Pu Qian Qiao Qiu Qu Ren Run Shan Shang Shen Si Song Su Sui Sun Tan Tang Tian Tu Wang Wu Xie Xiong Xu Yang Yao Ye Yin Zeng Zhang Zhao Zheng Zhou Zhu Zhuang Zou
% PERCENT cents percent
approvals bonus cash concessions cooperatives credit disbursements dividends donations earnings emoluments entitlements expenditure expenditures fund funding funds grants income incomes inflation lending liquidity loan loans mortgage mortgages overhead payroll pension pensions portfolio profits protectionism quotas receipts receivables remittances remuneration rent rents returns revenue revenues salaries salary savings spending subscription subsidies subsidy surplus surpluses tax taxation taxes tonnage tuition turnover wage wages
Abby Abigail Agnes Alexandra Alice Amanda Amy Andrea Angela Ann Anna Anne Annette Becky Beth Betsy Bonnie Brenda Carla Carol Carole Caroline Carolyn Carrie Catherine Cathy Cheryl Christina Christine Cindy Claire Clare Claudia Colleen Cristina Cynthia Danielle Daphne Dawn Debbie Deborah Denise Diane Dina Dolores Donna Doris Edna Eileen Elaine Eleanor Elena Elisabeth Ellen Emily Erica Erin Esther Evelyn Felicia Felicity Flora Frances Gail Gertrude Gillian Gina Ginger Gladys Gloria Gwen Harriet Heather Helen Hilary Irene Isabel Jane Janice Jeanne Jennifer Jenny Jessica Jo Joan Joanna Joanne Jodie Josie Judith Judy Julia Julie Karen Kate Katherine Kathleen Kathryn Kathy Katie Kimberly Kirsten Kristen Kristin Laura Laurie Leah Lena Lillian Linda Lisa Liz Liza Lois Loretta Lori Lorraine Louise Lynne Marcia Margaret Maria Marian Marianne Marilyn Marjorie Marsha Mary Maureen Meg Melanie Melinda Melissa Merle Michele Michelle Miriam Molly Nan Nancy Naomi Natalie Nina Nora Norma Olivia Pam Pamela Patricia Patti Paula Pauline Peggy Phyllis Rachel Rebecca Regina Renee Rita Roberta Rosemary Sabrina Sally Samantha Sarah Selena Sheila Shelley Sherry Shirley Sonia Stacy Stephanie Sue Susanne Suzanne Suzy Sylvia Tammy Teresa Teri Terri Theresa Tina Toni Tracey Ursula Valerie Vanessa Veronica Vicki Vivian Wendy Yolanda Yvonne
almonds apple apples asparagus avocado bacon bananas barley basil bean beans beets berries berry boneless broccoli cabbage carrot carrots celery cherries cherry chile chiles chili chilies chives cilantro citrus cranberries cranberry cucumber cucumbers dill doughnuts egg eggplant eggs elk evergreen fennel figs flowers fruit fruits garlic ginger grapefruit grasses herb herbs jalapeno Jell-O lemon lemons lettuce lime lions macaroni mango maple melon mint mozzarella mushrooms oak oaks olives onion onions orange oranges orchids oregano oyster parsley pasta pastries pea peach peaches peanuts pear pears peas pecan pecans perennials pickles pine pineapple pines plum pumpkin pumpkins raspberries raspberry rice rosemary roses sage salsa scallions scallops seasonings seaweed shallots shrimp shrubs spaghetti spices spinach strawberries strawberry thyme tomato tomatoes truffles tulips turtles walnut walnuts watermelon wildflowers zucchini
mid-April mid-August mid-December mid-February mid-January mid-July mid-June mid-March mid-May mid-November mid-October mid-September mid-afternoon midafternoon midmorning midsummer

Table 4: Examples of clusters

ing 13 words and the largest cluster containing 20,396 words. Table 4 shows some examples of the generated clusters. For each cluster we list all words occurring more than 1,000 times in the corpus.

7 Conclusion

In this paper, we have introduced an efficient, distributed clustering algorithm for obtaining word classifications for predictive class-based language models with which we were able to use billions of tokens of training data to obtain classifications for millions of words in relatively short amounts of time.

The experiments presented show that predictive class-based models trained using the obtained word classifications can improve the quality of a state-of-the-art machine translation system as indicated by the BLEU score in both translation tasks. When using predictive class-based models in combination with a word-based language model trained on very large amounts of data, the improvements continue to be statistically significant on the *test* and *nist06* sets. We conclude that even despite the large amounts of data used to train the large word-based model in our second experiment, class-based language models are still an effective tool to ease the effects of data sparsity.

We furthermore expect to be able to increase the gains resulting from using class-based models by using more sophisticated techniques for combining them with word-based models such as linear interpolations of word- and class-based models with coefficients depending on the frequency of the history.

Another interesting direction of further research is to evaluate the use of the presented clustering technique for language model size reduction.

References

Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. 2007. Large language models in machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing and on Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic.

Peter F. Brown, Vincent J. Della Pietra, Peter V. de Souza, Jennifer C. Lai, and Robert L. Mercer. 1990. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Jeffrey Dean and Sanjay Ghemawat. 2004. MapReduce: Simplified data processing on large clusters. In *Proceedings of the Sixth Symposium on Operating System Design and Implementation (OSDI-04)*, San Francisco, CA, USA.

Ahmad Emami and Frederick Jelinek. 2005. Random clusterings for language modeling. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Philadelphia, PA, USA.

Joshua Goodman and Jianfeng Gao. 2000. Language model size reduction by pruning and clustering. In *Proceedings of the IEEE International Conference on Spoken Language Processing (ICSLP)*, Beijing, China.

Joshua Goodman. 2000. A bit of progress in language modeling. Technical report, Microsoft Research.

Reinherd Kneser and Hermann Ney. 1993. Improved clustering techniques for class-based statistical language modelling. In *Proceedings of the 3rd European Conference on Speech Communication and Technology*, pages 973–976, Berlin, Germany.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Barcelona, Spain.

Sven Martin, Jörg Liermann, and Hermann Ney. 1998. Algorithms for bigram and trigram word clustering. *Speech Communication*, 24:19–37.

Eric W. Noreen. 1989. *Computer-Intensive Methods for Testing Hypotheses*. John Wiley & Sons, New York.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 295–302, Philadelphia, PA, USA.

Franz Josef Och. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo, Japan.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 311–318, Philadelphia, PA, USA.

Martin Raab. 2006. Language model techniques in machine translation. Master's thesis, Universität Karlsruhe / Carnegie Mellon University.

E. W. D. Whittaker and P. C. Woodland. 2001. Efficient class-based language modelling for very large vocabularies. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 545–548, Salt Lake City, UT, USA.