Adaptive Language and Translation Models for Interactive Machine Translation

Laurent Nepveu, Guy Lapalme
Philippe Langlais
RALI/DIRO - Université de Montréal,
C.P. 6128, succursale Centre-ville
Montréal, Québec, Canada H3C 3J7
{nepveul,lapalme,felipe}
@iro.umontreal.ca

George Foster
Language Technologies Research Centre
National Research Council Canada
A-1330, 101 rue Saint-Jean Bosco,
Gatineau, Québec, Canada K1A 0R6
George.Foster@nrc-cnrc.gc.ca

Abstract

We describe experiments carried out with adaptive language and translation models in the context of an interactive computer-assisted translation program. We developed cache-based language models which were then extended to the bilingual case for a cache-based translation model. We present the improvements we obtained in two contexts: in a theoretical setting, we achieved a drop in perplexity for the new models and, in a more practical situation simulating a user working with the system, we showed that fewer keystrokes would be needed to enter a translation.

1 Introduction

Cache-based language models were introduced by Kuhn and de Mori (1990) for the dynamic adaptation of speech language models. These models, inspired by the memory caches on modern computer architectures, are motivated by the principle of locality which states that a program tends to repeatedly use memory cells that are physically close. Similarly, when speaking or writing, humans tend to use the same words and phrase constructs from paragraph to paragraph and from sentence to sentence. This leads us to believe that, when processing a document, the part of a document that is already processed (e.g. for speech recognition, translation or text prediction) gives us very useful information for future processing in the same document or in other related documents.

A cache-based language model is a language model to which is added a smaller model trained only on the history of the document being processed. The history is usually the last N words or sentences seen in the document.

Kuhn and de Mori (1990) obtained a drop in perplexity of nearly 68% when adding an unigram POS (*part-of-speech*) cache on a 3g-gram model. Martin and al. (1997) obtained a drop of nearly 21% when adding a bigram cache to a trigram model. Clarkson and Robertson (1997) also obtained similar results with an exponentially decaying unigram cache.

The major problem with these theoretical results is that they assume the correctness of the material entering the cache. In practice, this assumption does not always hold, and so a cache can sometimes do more harm than good.

1.1 Interactive translation context

Over the last few years, an interactive machine translation (IMT) system (Foster et al., 2002) has been developed which, as the translator is typing, suggests word and phrase completions that the user can accept or ignore. The system uses a translation engine to propose the words or phrases which it judges the most probable to be immediately typed. This engine includes a translation model (TM) and a language model (LM) used jointly to produce proposals that are appropriate translations of source words and plausible completions of the current text in the target language. The translator remains in control of the translation because what is typed by the user is taken as a constraint to which the model must continually adapt its completions. Experiments have shown that the use of this system can save about 50% of the keystrokes needed for entering a translation. As the translation and language models are built only once, before the user starts to work with the system, the translator is often forced to repeatedly correct similar suggestions from the system.

The interactive nature of this setup made us believe that it is a good prospect for dynamic adaptive modeling. If the dynamic nature of the system can be disadvantageous for static language and translation models, it is an incomparable advantage for a cache based approach because human correction intervenes *before* words go in the cache. As the translator is using the system to correctly enter his translation progressively, we can expect the theoretical results presented in the literature to be obtainable in practice in the IMT context.

The first advantage of dynamic adaptation would be to help the translation engine make better predic-

tions, but it has a further *psychological* advantage: as the translator works and potentially corrects the proposals of the engine, the user would feel that the software is learning from its errors.

The next section describes the models currently embedded within our IMT prototype. Section 3 describes the cache-based adaptation we performed on the target language model. In section 4, we present the different types of adaptations we performed on the translation model. Section 5 then puts the results in the context of our IMT application. Section 6 discusses the implications of our experiments and suggests some improvements that could be made to the system.

2 Current IMT models

The word-based translation model embedded within the IMT system has been designed by Foster (2000). It is a Maximum Entropy/Minimum Divergence (MEMD) translation model (Berger et al., 1996), which mimics the parameters of the IBM model 2 (Brown et al., 1993) within a log-linear setting.

The resulting model (named MDI2B) is of the following form, where $\mathbf{h}$ is the current target text, $\mathbf{s}$ the source sentence being translated, s a particular word in $\mathbf{s}$ and w the next word to be predicted:

$$p(w|\mathbf{h},\mathbf{s}) = \frac{q(w|\mathbf{h})\exp(\sum_{s\in\mathbf{s}}\alpha_{sw}+\beta_{AB})}{Z(\mathbf{h},\mathbf{s})} \quad (1)$$

The q distribution represents the prior knowledge that we have about the true distribution and is modeled by an interpolated trigram in this study. The α coefficients are the familiar transfer or lexical parameters, and the β ones can be understood as their position dependent correction. Z is a normalizing factor, the sum of the numerator for every w in the target vocabulary.

Our baseline model used an interpolated trigram of the following form as the q distribution:

$$\begin{aligned} p(w|\mathbf{h}) &= \lambda_1(w_{i-2}w_{i-1}) \times p_{tri}(w_i|w_{i-2}w_{i-1}) \\ &+ \lambda_2(w_{i-2}w_{i-1}) \times p_{bi}(w_i|w_{i-1}) \\ &+ \lambda_3(w_{i-2}w_{i-1}) \times p_{uni}(w_i) \\ &+ \lambda_4(w_{i-2}w_{i-1}) \times \tfrac{1}{|V|+1} \end{aligned}$$

where $\lambda_1(w_{i-2}w_{i-1}) + \lambda_2(w_{i-2}w_{i-1}) + \lambda_3(w_{i-2}w_{i-1}) + \lambda_4(w_{i-2}w_{i-1}) = 1$ and $|V|+1$ is the size of the event space (including a special *unknown* word).

As mentioned above, the MDI2B model is closely related to the IBM2 model (Brown et al., 1988). It contains two classes of features: word pair features and positional features. The word pair feature functions are defined as follows:

$$f_{st}(w,\mathbf{h},\mathbf{s}) = \begin{cases} 1 & \text{if s} \in \mathbf{s} \text{ and } t = w \\ 0 & \text{otherwise} \end{cases}$$

This function is *on* if the predicted word is t and s is in the current source sentence. Each feature f_{st} has a corresponding weight α_{st} (for brevity, this is defined to be 0 in equation 1 if the pair s,t is not included in the model).

The positional feature functions are defined as follows:

$$f_{A,B}(w,i,\mathbf{s}) = \sum_{j=1}^{J} \delta[(i,j,J) \in A \wedge (s_j,w) \in B \wedge j = \hat{\jmath}_{s_j}]$$

where $\delta[X]$ is 1 if X is true, otherwise 0; and $\hat{\jmath}_{s_j}$ is the position of the occurrence of s_j that is *closest* to i according to an IBM2 model. A is a class that groups positional (i,j,J) configurations having similar IBM2 alignment probabilities, in order to reduce data sparseness. B is a class of word pairs having similar weights α_{st}. Its purpose is to simulate the way IBM2 alignment probabilities modulate IBM1 word-pair probabilities, by allowing the value of the positional feature weight to depend on the *magnitude* of the corresponding word-pair weight. As with the word pair features, each $f_{A,B}$ has a corresponding weight β_{AB}.

Since feature selection is applied at training time in order to improve speed, avoid overfitting, and keep the model compact, the summation in the exponential term in (1) is only carried out over the set of *active* pairs maintained by the model and not over all pairs as might be inferred from the formulation.

To give an example of how the model works, if the source sentence is *the fruit I am eating is a banana* and we are predicting the word *banane* following the target words: *Le fruit que je mange est une*, the active pairs involving *banana* would be (*fruit*, *banana*) and (*banane*, *banana*) since, of all the pairs (s,t) they would be the only ones kept by the feature selection algorithm[1]. The probability of *banane* would therefore depend on the weights of those two pairs, along with position weights which capture the relative proximity of the words involved.

3 Language model adaptation

We implemented a first monolingual dynamic adaptation of this model by inserting a cache component in its reference distribution, thus only affecting the q distribution. We obtained similar results

[1] See (Foster, 2000) for the description of this algorithm.

as for classical ngram models: the unigram cache model proved to be less efficient than the bigram one, and the trigram cache suffered from sparsity. We also tested a model where we interpolated the three cache models to gain information from each of the unigram, bigram, and trigram cache models. For completeness, this generalized model is described in equation 2 under the usual constraints that $\sum_i \lambda_i(h) = 1$ for all h.

$$\begin{aligned} p(w|\mathbf{h}) &= \lambda_1(\mathbf{h}) \times p_{tri}(w_i|w_{i-2}w_{i-1}) \\ &+ \lambda_2(\mathbf{h}) \times p_{bi}(w_i|w_{i-1}) \\ &+ \lambda_3(\mathbf{h}) \times p_{uni}(w_i) \\ &+ \lambda_4(\mathbf{h}) \times \tfrac{1}{|V|+1} \\ &+ \lambda_5(\mathbf{h}) \times p_{tric}(w_i|w_{i-2}w_{i-1}) \\ &+ \lambda_6(\mathbf{h}) \times p_{bic}(w_i|w_{i-1}) \\ &+ \lambda_7(\mathbf{h}) \times p_{unic}(w_i) \end{aligned} \quad (2)$$

Those models were trained from splits of the Canadian Hansard corpus. The base ngram model was estimated with a 30M word split of the corpus. The weighting coefficients of both the base trigram and the cache models were estimated with an EM algorithm trained with 1M words.

We tested our models, translating from English to French, on two corpora of different types: the first one `hansard` is a document taken from the same large corpus that was used for training (the testing and training corpora were exclusive splits). The second one `sniper`, which describes the job of a sniper, is from another domain characterized by lexical and phrasal constructions very different from those used to estimate the probabilities of our models.

Table 1 shows the perplexity on the `hansard` and the `sniper` corpora. Preliminary experiments led us to two sizes of cache which seemed promising: 2000 and 5000 corresponding to the last 2000 and 5000 words seen during the processing of a document. The **BI** column gives the results of the bigram cache model and the **1+2+3** gives the results of the interpolated cache model which included the unigram, bigram and trigram cache.

The results show that our models improve the base static model by 5% on documents supposedly *well known* by the models and by more that 52% on documents that are *unknown* to the model. Section 5 puts these results in the perspective of our actual IMT system. Note that he addition of a cache component to a language model involves negligible extra training time.

Taille	**BI**	Δ	**1+2+3**	Δ
base	`hansard`=17.6584			
2000	16.937	-4.1%	16.840	-4.6%
5000	16.903	-4.3%	16.777	-5.0%
base	`sniper`=135.808			
2000	73.936	-45.6%	67.780	-50.1%
5000	70.514	-48.1%	64.204	-52.7%

Table 1: Perplexities of the MDI2B model with a cache component included in the reference distribution on the `hansard` and `sniper` corpora.

4 Translation model adaptation

With those excellent results in mind, we extended the idea of dynamic adaptation to the bilingual case which, to our knowledge, has never been tried before.

We developed a model called MDI2BCache which is a MDI2B model to which we added a cache component based on word pairs. Recall that, when predicting a word w at a certain point in a document, the probability depends on the weights of the pairs (s, w) for each active word s in the current source sentence. As the prediction of the words of the document goes on, our model keeps in a cache each active pair used for the prediction of each word. In the example above, if the translator accepts the word *banane*, then the two pairs (*fruit*, *banana*) and (*banane*, *banana*) will be added to the cache.

We added a new feature to the MEMD model to take into account the presence of a certain pair in the recent history of the processed document:

$$f_{cache\ st}(w, \mathbf{h}, \mathbf{s}) = \begin{cases} 1 & if \begin{cases} s \in \mathbf{s}, \\ t = w, \\ (s,t) \in cache \\ \alpha_{st} > p \end{cases} \\ 0 & otherwise \end{cases}$$

We added a threshold value p to the feature function because while analyzing the pair weights, we discovered that low weight pairs are usually pairs of utility words such as conjunctions and punctuation. We also came to the conclusion that they are not the kind of words we want to have in the cache, since their presence in a sentence implies little about their presence in the next.

The resulting model is of the form:

$$p(w|\mathbf{h}, \mathbf{s}) = \frac{q(w|\mathbf{h}) exp(\sum_{s \in \mathbf{s}} \alpha_{sw} + \beta_{AB} + \gamma_{sw})}{Z(\mathbf{h}, \mathbf{s})}$$

Thus, every $f_{cache\ sw}$ has a corresponding weight γ_{sw} for the calculation of the probability of w.

Size	0.3	Δ	0.5	Δ	0.7	Δ
base	One feature weight, no Viterbi				orig perp=17.6584	
1000	17.5676	-0.51%	17.5756	-0.47%	17.5983	-0.34%
2000	17.5698	-0.50%	17.5766	-0.46%	17.5976	-0.34%
5000	17.5743	-0.48%	17.5776	-0.46%	17.5965	-0.35%
10000	17.5777	-0.46%	17.5791	-0.45%	17.5962	-0.35%
base	One feature weight per pair, no Viterbi				orig perp=17.6584	
1000	17.5817	-0.43%	17.5858	-0.41%	17.6065	-0.29%
2000	17.5933	-0.37%	17.5918	-0.38%	17.6061	-0.30%
5000	17.5849	-0.42%	17.5874	-0.40%	17.6076	-0.29%
10000	17.5890	-0.39%	17.5891	-0.39%	17.6069	-0.29%
base	One feature weight, Viterbi				orig perp=17.6584	
1000	17.5602	-0.56%	17.5697	-0.50%	17.5940	-0.36%
2000	17.5676	-0.51%	17.5695	-0.50%	17.5896	-0.39%
5000	17.5614	-0.55%	17.5687	-0.51%	17.5925	-0.37%
10000	17.5650	-0.53%	17.5687	-0.51%	17.5906	-0.38%

Table 2: MDI2BCache test perplexities. One feature weight, Viterbi alignment version.

4.1 Number of cache features

We implemented two versions of the model, one in which we estimated only one cache feature weight for the whole model and another in which we estimated one cache feature weight for every word pair in the model.

The first model is simpler and is easier to estimate. The assumption is made that every pair in the model has the same tendency to repeat itself.

The second model doubles the number of word-pair parameters compared to MDI2B, and thus leads to a linear increase in training time. Extra training time is negligible in the first model.

4.2 Word alignment

One of the main difficulties of automatic MT is determining which source word(s) translate to which target word(s). It is very difficult to do this task automatically, in part because it is also very difficult manually. If a pair of sentences are given to 10 translators for alignment, the results would likely not be identical in all cases. As it is nearly impossible to determine such an alignment, most translation models consider every source word to have an effect on the translation of every target word.

This difficulty shows up in our cache-based model. When adding word pairs to the cache, we ideally would like to add only word pairs that were really in a translation relation in the given sentence.

This is why we also implemented a version of our model in which a word alignment is first carried out in order to select good pairs to be added to the cache. For this purpose, we computed a Viterbi alignment based on an IBM model 2. This results in a subset of the *good* active pairs to be added to the cache. The Viterbi algorithm gives us a higher confidence level that the pair of words added to the cache were really in a translation relation. But it can also lead to word pairs not added to the cache that should have been added.

4.3 Results

Table 2 shows the results of the different configurations of the MDI2BCache model. For every configuration we trained and tested on splits of the Canadian Hansard with threshold values of 0.3, 0.5, and 0.7 and cache sizes of 1000, 2000, 5000, and 10000. The top of the table is the version of the model with only one feature weight without Viterbi alignment. The middle of the table is the version with one feature weight per word pair without Viterbi alignment. Finally, the bottom is for the version with only one feature weight and a Viterbi alignment made prior to adding pairs to the cache.

Threshold values of 0.3, 0.5, and 0.7 led to 75%, 50%, and 25% of the pairs considered for addition to the cache respectively. The results show that the threshold values of 0.5 and 0.7 are removing too many pairs. The best results are obtained with a threshold of 0.3 in all tests. Since the number of pairs kept in the model appears to vary in proportion to the threshold value, we did not consider it necessary to use an automatic search algorithm to find an optimal threshold value. The gain in performance would have been negligible.

The results also show that having one feature weight per word pair leads to lower results. This can be explained by the fact that it is much more

Size	0.3	Δ	0.5	Δ
base	MDI2B=135.808			
1000	132.865	-2.17%	132.751	-2.25%
2000	132.771	-2.23%	132.752	-2.25%
5000	132.733	-2.26%	132.628	-2.34%
10000	132.997	-2.07%	132.674	-2.31%

Table 3: MDI2BCache test perplexities. One feature weight, Viterbi alignment version. Sniper test

difficult to estimate a weight for every pair that one weight for all pairs. Since we use only thousands of words in the cache, the training process suffers from a poor data representation.

The Viterbi alignment seems to be helping the models. The best results are obtained with the version of our model with Viterbi alignment. However, this gives only a 0.56% percent drop in perplexity.

We then tested our best configuration on the `sniper` corpus. Table 3 shows the results. We dropped threshold value 0.7 and tested only the model with only one feature weight and a Viterbi alignment.

Results show that our bilingual cache model shows improvement (four times higher) in drop of perplexity when used on documents very different from the training corpus. In general, results give lower perplexity than our base model showing that the bilingual cache is helpful to the model, but the results are not as good as that the ones obtained in the unilingual case. Section 6 discusses these results further.

5 Evaluation of IMT

As stated earlier, drops in perplexity are theoretical results that have been obtained previously in the case of unilingual dynamic adaptation but for which a corresponding level of practical success was rarely attained because of the cache correctness problem. To show that the interactive nature of our assisted-translation application can really benefit from dynamic adaptation, we tested our models in a more realistic translation context. This test consists of simulating a translator using the IMT system as it proposes words and phrases and accepting, correcting or rejecting the proposals by trying to reproduce a given target translation (Foster et al., 2002). The metric used is the percentage of keystrokes saved by the use of the system instead of having to type directly all the target text.

For these simulations, we used only a 10K word split of the `hansard` and of the `sniper` corpus. The reason is that the IMT application potentially proposes new completions after every character typed by the user. For a 10K word document, it needs to search about 1 million times for high probability words and phrases. This leads to relatively long simulation times, even though predictions are made at real time speeds.

Taille	BI	Δ	1+2+3	Δ
base	`hansard`=27.435			
2000	27.784	+1.3%	27.719	+1.0%
5000	27.837	+1.5%	27.821	+1.4%
base	`sniper`=9.686			
2000	11.404	+15.1%	11.294	+14.2%
5000	11.498	+15.8%	11.623	+16.7%

Table 4: Saved keystrokes raises for the MDI2B model with cache component in the reference distribution on the `hansard` and `sniper` corpora.

	0.3	Δ
base	`hansard`=27.4358	
1000	27.557	+0.44%
2000	27.531	+0.35%
5000	27.488	+0.18%
10000	27.468	+0.12%
base	`sniper`=9.686	
1000	9.896	+2.17%
2000	10.023	+3.48%
5000	9.983	+3.07%
10000	9.957	+2.80%

Table 5: Saved keystrokes raises for the MDI2BCache model with only one feature weight and Viterbi alignment on the `hansard` and `sniper` corpora.

Table 4 shows the results obtained with the MDI2B model to which we added a cache component for the reference interpolated trigram distribution.

We can see that the saved keystroke percentages are proportional to the perplexity drops reported in section 3. The use of our models raises the saved keystrokes by nearly 1.5% in the case of *well known* documents and by nearly 17% in the case of very different documents. These are very interesting results for a potential professional use of TransType.

Table 5 shows an increase in the number of saved keystrokes: 0.44% on the `hansard` and 3.5% on the `sniper` corpora. Once again, the results are not as impressive as the ones obtained for the monolingual dynamic adaptation case.

6 Discussion

The results presented in section 3 on language model adaptation confirmed what had been reported in the literature: adding a cache component to a language model leads to a drop in perplexity. Moreover, we were able to demonstrate that using a cache-based language model inside a translation model leads to better performance for the whole translation model. We obtained drops in perplexity of 5% on a corpus of the same type as the training corpus and of 50% on a different one. These theoretical results lead to very good practical results. We were able to increase the saved keystroke percentage by 1.5% on the similar corpus as the training and by nearly 17% on the different corpus. These results confirm our hypothesis that dynamic adaptation with cache-based language model can be useful in the context of IMT, particularly for new types of texts.

Results presented in section 4 on translation model adaptation show that our approach has led to drops in perplexity although not as high as we would have hoped. To understand these disappointing results, we analyzed the content of the cache for different configurations of our MDI2BCache model.

base	**0.3**	**viterbi + 0.3**
(*is*,*qu'*)	(*to*,*afin*)	(*offence*,*crime*)
(.,*sa*)	(*was*,*a*)	(*was*,*été*)
(*this*,,)	(*UNK*,*UNK*)	(*very*,*très*)
(*all*,*toutes*)	(*piece*,*législative*)	(*today*,*aujourd'hui*)
(*have*,*du*)	(*this*,*ce*)	(*jobs*,*emploi*)
(*the*,*pour*)	(*per*,*100*)	(*concern*,*inquiétude*)
(*on*,*du*)	(*that*,*soient*)	(*skin*,*peau*)
(*of*,*un*)	(,,,)	(*there*,*y*)
(*we*,*nous*)	(*?*,*il*)	(*government*,*le*)
(*the*,*du*)	(*any*,*tout*)	(*an*,*un*)
18	68	86

Table 6: Cache sampling of different configurations of MDI2BCache model.

Table 6 shows the results of our sampling. We tested three model configurations. The first one, in the first column, was the base MDI2BCache model which adds all active pairs to the cache. The second configuration, in the second column, was a threshold value of 0.3 that brings about 75% of the pairs being added to the cache. The last configuration was a model with threshold value of 0.3 and a Viterbi alignment made prior to the addition of pairs in the cache. The three model configuration were with only one feature weight. For all three configurations, we took a sample of 10 pairs (shown in table 6) and a sample of 100 pairs. With the second sample, we manually analyzed each pair and counted the number of pairs (shown in the last row of the table) we believed were useful for the model (words that are occasionally translations of one another).

The results obtained in section 4 seem to agree with the current analysis. From left to right in the table, the pairs seem to contain more information and to be more appropriate additions to the cache. The configuration with Viterbi alignment which contains 86 good pairs clearly seems to be the configuration with the most interesting pairs.

The problem with such a cache-based translation model seem to be similar to the balance between precision and recall in information retrieval. On one hand, we want to add in the cache every word pair in which the two words are in translation relation in the text. We further want to add *only* the pairs in which the two words are really in translation relation in the text. It seems that with our base model, we add most of the good pairs, but also a lot of bad ones. With the Viterbi alignment and a threshold value of 0.3, most of the pairs added are good ones, but we are probably missing a number of other appropriate ones. This comes back to the task of word alignment, which is a very difficult task for computers (Mihalcea and Pedersen, 2003).

Moreover, we would want to add in the cache only those words for which more than one translation is possible. For example, the pair (*today*, *aujourd'hui*), though it is a very useful pair for the base model, is unlikely to help when added to the cache. The reason is simple: they are two words that are always translations of one another, so the model will have no problem predicting them. This ideal of precision and recall and of useful pairs in the cache is obtained by our model with threshold of 0.3, a Viterbi alignment and a cache size of 1000.

One disadvantage of our bilingual adaptive model is the way it handles unknown words. In the cache-based language model, the unknown words were dealt with *normally*, i.e. they were added to the cache and given a certain probability afterwards. So, if an unknown word was seen in a certain sentence and then later on, it would receive a probability mass of its own but not the one given to any unknown word. By having its own probability mass due to its presence in the cache, such previously unknown word can be predicted by the model. In the case of our MDI2BCache model, because we have not yet implemented an algorithm for guessing the translations of unknown words, they are simply represented within the model as UNK words, which means that the model never learns them.

The results obtained with the `sniper` corpus shows us that dynamic adaptation is also more helpful for documents that are little *known* to the model in the bilingual context. The results are four times better on the `sniper` corpus than on the Hansard testing corpus.

Once again for the bilingual case, the practical test results in the number of saved keystrokes agree with the theoretical results of drops in perplexity. This result shows that bilingual dynamic adaptation also can be implemented in a practical context and obtain results similar to the theoretical results.

All things considered, we believe that a cache-based translation model shows a great potential for bilingual adaptation and that greater perplexity drops and keystroke savings could be obtained by either reengineering the model or by improving the MDI2BCache model.

6.1 Key improvements to the model

Following the analysis of the results obtained by our model, we have pointed out some key improvements that the model would need in order to get better results. In this list we focus on ways of improving adaptation strategies for the current model, omitting other obvious enhancements such as adding phrase translations.

Unknown word processing Learning new words would be a very important feature to add to the model and would lead to better results. We did not incorporate the processing of unknown words in the MDI2BCache because the structure of model did not lend itself to this addition. Especially with documents such as the `sniper` corpus, we believe that this could be a key improvement for a dynamic adaptive model.

Better alignment As mentioned before, the ultimate goal for our cache is that it contains only the pairs present in the *perfect* alignment. Better performance from the alignment would lead to pairs in the cache closer to this ideal. In this study we computed Viterbi alignments from an IBM model 2, because it is very efficient to compute and also because for training MDI2B, we do use the IBM model 2. We could consider also more advanced word alignment models (Och and Ney, 2000; Lin and Cherry, 2003; Moore, 2001). To keep the alignment model simple, we could still use an IBM model 2, but with the compositionality constraint that has been shown to give better word alignment than the Viterbi one (Simard and Langlais, 2003).

Feature weights We implemented two versions of our model: one with only one feature weight and another with one feature weight for each word pair. The second model suffered from poor data representation and our training algorithm wasn't able to estimate good cache feature weights. We think that creating classes of word pairs, such as it was done for positional alignment features, would lead to better results. It would enable the model to take into account the tendency that a pair has to repeat itself in a document.

Relative weighting Another key improvement is that changes to word-pair weights should be relative to each source word. For example, if (*house*, *maison*) is a pair in the cache, we would like to favour *maison* over possible alternatives such as *chambre* as a translation of *house*. In the existing model this is done by boosting the weight on (*house*,*maison*), which has the undesirable side-effect of making *maison* more important in the model than translations of other source words in the current sentence which have not appeared in the cache. One way of eliminating this behaviour would be to learn negative weights on alternatives like (*house*,*chambre*) which do not appear in the cache.

We believe these improvements would better show the potential of bilingual dynamic adaptation.

7 Conclusion

We have presented dynamic adaptive translation models using cache-based implementations. We have shown that monolingual dynamic adaptive models exhibit good theoretical performance in a bilingual translation context. We observed that these theoretical results carry over to practical gains in the context of an IMT application.

We have developed bilingual dynamic adaptation through a cache-based translation model. Our results show the potential of bilingual dynamic adaptation. We have given explanations about why the results obtained are not as high as hoped and presented some key improvements that should be made to our model or should be taken into account in the development of a new model.

We believe that this study reveals the potential for adaptive interactive machine translation system and we hope to read similar reports for other implementations of the same interactive scenario *e.g.* (Och et al., 2003).

References

Adam L. Berger, Stephen A. Della Pietra, and Vincent J. Della Pietra. 1996. A Maximum Entropy approach to Natural Language Processing. *Computational Linguistics*, 22(1):39–71.

Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Fredrick Jelinek, Robert L. Mercer, and Paul Roossin. 1988. A statistical approach to language translation. In *Proceedings of the International Conference on Computational Linguistics (COLING)*, pages 71–76, Budapest, Hungary, August.

Peter F. Brown, Stephen A. Della Pietra, Vincent Della J. Pietra, and Robert L. Mercer. 1993. The mathematics of Machine Translation: Parameter estimation. *Computational Linguistics*, 19(2):263–312, June.

P.R. Clarkson and P.R. Robertson. 1997. Language model adaptation using mixtures and an exponentially decaying cache. In *IEEE Int. Conference on Acoustics, Speech, and Signal Processing*, Munich.

George Foster, Philippe Langlais, and Guy Lapalme. 2002. User-friendly text prediction for translators. In *2002 Conference on Empirical Methods in Natural Language Processing (EMNLP 2002)*, Philadelphia, July. TT2 TransType2.

George Foster. 2000. A Maximum Entropy / Minimum Divergence translation model. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 37–42, Hong Kong, October.

Roland Kuhn and Renato De Mori. 1990. A cache-based natural language model for speech recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 12(6):570–583, June.

Dekang Lin and Colin Cherry. 2003. Proalign: Shared task system description. In *NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 11–14, Edmonton Canada, May 31. TT2.

S.C. Martin, J. Liermann, and H. Ney. 1997. Adaptative topic-dependent language modelling using word-based varigrams. In *Eurospeech*.

Rada Mihalcea and Ted Pedersen. 2003. An evaluation exercise for word alignment. In Rada Mihalcea and Ted Pedersen, editors, *HLT-NAACL 2003 Workshop: Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 1–10, Edmonton, Alberta, Canada, May 31. Association for Computational Linguistics.

Robert C. Moore. 2001. Towards a simple and accurate statistical approach to learning translation relationships among words. In *Workshop on Data-driven Machine Translation, 39th Annual Meeting and 10th Conference of the European Chapter*, pages 79–86, Toulouse, France. Association for Computational Linguistics.

Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 440–447, Hong Kong, October.

F.J. Och, R. Zens, and H. Ney. 2003. Efficient search for interactive statistical machine translation. In *Proceedings of the 10th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 387–393, Budapest, Hungary, April. TT2.

Michel Simard and Philippe Langlais. 2003. Statistical translation alignment with compositionality constraints. In *NAACL 2003 Workshop on Building and Using Parallel Texts: Data Driven Machine Translation and Beyond*, pages 19–22, Edmonton Canada, May 31. TT2.