On LM Heuristics for the Cube Growing Algorithm

David Vilar and Hermann Ney
Lehrstuhl für Informatik 6
RWTH Aachen University
52056 Aachen, Germany
{vilar,ney}@informatik.rwth-aachen.de

Abstract

Current approaches to statistical machine translation try to incorporate more structure into the translation process by including explicit syntactic information in form of a formal grammar (with a possible, but not necessary, correspondence to a linguistic motivated grammar). These more structured models incur into an increased generation cost, and efficient algorithms must be developed. In this paper we concentrate on the *cube growing* algorithm, a lazy version of the cube grow algorithm. The efficiency of this algorithm depends on a heuristic for language model computation, which is only scarcely discussed in the original paper. In this paper we investigate the effect of this heuristic on translation performance and efficiency and propose a new heuristic which efficiently decreases memory requirements and computation time, while maintaining translation performance.

1 Introduction

In the last decade, the phrase-based approach to machine translation has been the de-facto standard for statistical machine translation (SMT) systems. The main reason was that it offered a great improvement in translation quality over its predecessors, the single-word based models. The model itself is also relatively simple and allows for efficient generation algorithms like for example beam search (see e.g. (Koehn, 2004)). This allowed the approach to scale to bigger tasks and it is still one of the most widely used models nowadays. The current trend in SMT, however, is to bring more information in the form of grammatical structures. There are mainly two possibilities, in the form of linguistically motivated grammars (e.g. (Marcu et al., 2006)) or just formal grammars, which do not need to have a linguistic equivalent (e.g. (Chiang, 2005)).

These more expressive models have associated a more difficult search problem, which normally involves a parsing process (usually a variation of the CYK algorithm) while incorporating the translation information. The inclusion of language model (LM) information replicates nodes in the parsing tree, which increases the cost of the generation process. And not to be underestimated, the number of rules the system has to deal with can be of one order of magnitude bigger than the standard phrase-based approach, depending on the model[1].

Therefore, new, efficient algorithms for translation with these richer models had to be developed. In this paper we will concentrate on the cube growing algorithm (Huang and Chiang, 2007), a lazy version of the cube pruning algorithm (Chiang, 2007). These algorithms represent the search space as an hypergraph and add language model scores as necessary.

The most time-consuming operation in the translation process is the LM score computation, especially when huge LMs are used. The cube growing algorithm follows an on-demand computation strategy and tries to minimize the number of LM scores that need to be computed. In order to minimize the number of search errors, while still maintaining computational efficiency, the algorithm depends on an (efficient) heuristic for these LM costs, which is only scarcely dis-

[1] Note that most of these models do not discard the (majority of) standard phrases, instead they add new rules to the phrase inventory.

Proceedings of the 13th Annual Conference of the EAMT, pages 242–249,
Barcelona, May 2009

cussed in the original paper.

In this work we investigate the originally proposed heuristic in terms of translation quality and computational cost and propose a new heuristic, which maintains translation performance while reducing memory requirements at no computation time expense. The main idea is to cluster the words in the target language into a reduced number of classes and to compute an optimistic LM score on these classes, a concept similar to the one presented in (Petrov et al., 2008). We focus our attention on the hierarchical phrase-based model proposed in (Chiang, 2007), but our findings may as well be applicable to other translation models.

This paper is structured as follows. Section 2 reviews the hierarchical phrase-based approach to SMT and Section 3 the cube growing algorithm. In Section 4 we present the requirements for the heuristics to be used in this algorithm. Section 5 presents our new heuristic. Experimental results are presented in Section 6 and discussed in Section 7. The paper concludes in Section 8.

2 The Hierarchical Phrase Based Approach

The hierarchical phrase-based approach can be considered as an extension of the standard phrase-based model. In this model we allow the phrases to have "gaps", i.e. we allow non-contiguous parts of the source sentence to be translated into possibly non-contiguous parts of the target sentence. The model can be formalized as a synchronous context-free grammar (Chiang, 2007). The bilingual rules are of the form

$$X \rightarrow \langle \gamma, \alpha, \sim \rangle \,, \tag{1}$$

where X is a non-terminal, γ and α are strings of terminals and non-terminals, and $\sim$ is a one-to-one correspondence between the non-terminals of α and γ.

Two examples of this kind of rules for the German-to-English translation direction are

$$X \rightarrow \langle \text{ich habe } X^{\sim 0} \text{ gesehen}, \text{I have seen } X^{\sim 0} \rangle$$
$$X \rightarrow \langle \text{im } X^{\sim 0} \text{ zu } X^{\sim 1}, \text{in order to } X^{\sim 1} X^{\sim 0} \rangle$$

where the indices in the non-terminals represent the correspondence between source and target "gaps". This model has the additional advantage that reordering is integrated as part of the model itself, as can be seen in the above examples.

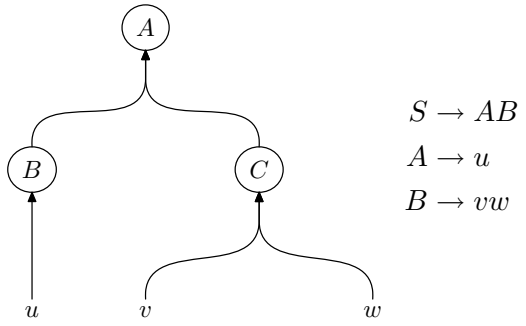


Figure 1: Example of an hypergraph corresponding to a grammar derivation. The hypergraph corresponds to the derivation of the string uvw using the grammar shown on the right.

The first step in the hierarchical phrase extraction is the same as for the phrased-based model. Having a set of initial phrases, we search for phrases which contain other smaller sub-phrases and produce a new phrase with gaps. In our system, we restricted the number of non-terminals for each hierarchical phrase to a maximum of two, which were also not allowed to be adjacent in the source side, and the gaps were allowed to have a maximum size of 10 words. The scores of the phrases are computed as relative frequencies.

3 The Cube Growing Algorithm

Starting point for the cube growing algorithm is a representation of the parsing space by means of an hypergraph. An hypergraph is an extension of the standard graph concept, where each (hyper-)node can have more than one predecessor. The hyperedges have thus an arity, which indicates the number of predecessors of the goal node. An example visualization of a simple derivation is shown in Figure 3. In our case the hypergraph is generated applying the CYK+ algorithm (Chappelier and Rajman, 1998) on the source sentence.

In the following, an informal description of the cube growing algorithm will be given, stressing the usage of heuristic function for the LM score computation. For a full detailed description, the reader is advised to consult (Huang and Chiang, 2007). In our description we will use the terms derivation and translation interchangeably. Remember that the hierarchical model is represented as a parallel grammar. Therefore, given a derivation of the source sentence, we can construct a corresponding derivation in the target language and thus obtain a translation. It is true that the same translation

(considered only at the word level) may arise from different source derivations. However, a strict distinction is not necessary in this context and would only clutter the text.

We will consider cost minimization in the following exposition, i.e. the best derivation is the one with a minimum cost, and costs are combined by adding them. The main procedure of the cube growing algorithm finds the n-th best derivation of a given node in an hypergraph. In order to do this, it recursively calls itself on the predecessor nodes, computing the necessary subderivations on demand. E.g. assume that the 10-th best derivation of an hypernode is formed by combining the 2nd-best and the 4-th best derivation of two predecessor hypernodes. In this case, only those 2 and 4 derivations of the predecessor nodes would have been computed[2].

If no LM score is taken into account, this computation can be carried out in an exact way. The different translation alternatives for one hyperedge can be sorted according to their costs. The derivations in the hypernodes with no predecessors (purely lexical rules), can thus be generated in a monotonic way. This allows, with a proper combination strategy, to generate the derivations in every hypernode in a cost-increasing order.

When including the LM score, however, the situation is different. The costs of the derivation are no longer simply the sum of the corresponding derivations in the predecessor hypernodes plus the cost of the grammar rule. Now we have to add the cost of the language model computed on the associated target language parts. The language model score is called a *combination cost*, as it is a cost that affects the combination of hypernodes. This score is costly to compute and dependent on all elements participating in the combination (predecessor hypernodes and translation rule). The effect of this is that the sorting strategy referred to above cannot longer guarantee that the generation of the derivations in an hypernode will proceed in a monotonic order.

In order to overcome this problem, we store the generated derivations in an intermediate buffer and we will only extract them from this buffer when we are confident that no better derivation can be found for the given hypernode. In the original paper it is shown that, if one can define an heuristic for the LM costs along an hyperedge, this technique will generate an exact n-best list of derivations.

The original paper proposes to compute an n-best list of translations without taking into account the LM scores, a so-called "-LM" parse (possibly taking into account recombination of hypotheses). Afterwards they compute the LM scores of these n-best derivations, and use these scores as the heuristic for the hyperedges involved in these derivations. The motivation behind this approach is that in the -LM pass, we will compute a hopefully representative portion of the needed derivations and thus, the best of these scores should act as an heuristic for the hyperedge. Note however, that there is no guarantee that the explored space will be big enough. If when taking the LM into account (the "+LM" parse) we need the heuristic for a hyperedge which was not computed in the -LM pass, we just take the LM score of the first-best derivation for this hyperedge.

Note that there is also an additional parameter for controlling the computational cost by limiting the number of derivation to be taken into account, i.e. the size of the intermediate buffer. This parameter can also have an effect when counteracting ill-formed heuristics.

4 Heuristic requirements

In order for the algorithm not to produce search errors, the heuristic must be optimistic (also called "acceptable"), that is, the costs given by the heuristic must be less than the actual cost. If this can be guaranteed, it can be shown that the search algorithm does not produce any search errors.

Another key issue for practical application is the necessity that the heuristic computation must be efficient. If too much time is spent on computing the heuristic, the gains of the lazy evaluation can be overcome by this computation time. In the extreme case, we could compute the LM cost of all possible combination at each hypernode, which will lead to an optimal heuristic. Of course this computation would be much more costly than the actual search using the cube growing algorithm.

In the case of the -LM heuristic, we can not guarantee its acceptability, as we cannot show that the hyperedges used in the -LM n-best computation will be reused in the +LM parse. In fact, the translations produced without language model differ much from the ones when the language model is taken into account, therefore it is not clear the

[2]By comparison, the cube pruning algorithm computes a fixed number of derivations at each hypernode.

adequacy of this heuristic. The efficiency can be controlled by varying the size of the n-best list, however small values of n can increase the risk of inappropriate heuristic values.

5 Coarse LM Heuristic

In this section we propose a new heuristic for the score computation of the members of the intermediate buffer. We first recall that, given an n-gram language model, the score of a word w given its context h (also called history) is given by the expression (Kneser and Ney, 1995)

$$p(w|h) = \begin{cases} \alpha(w|h) & \text{if } N(h,w) > 0 \\ \gamma(h)\alpha(w|\overline{h}) & \text{if } N(h,w) = 0 \end{cases} \quad (2)$$

where $N(h,w)$ corresponds to the word-history count in the training corpus, $\alpha(w|h)$ is the (discounted) relative frequency of the word-history pair, $\gamma(h)$ is a back-off weight, which also ensures a proper normalization of the probability distribution and $\overline{h}$ is a generalized history, that is, h with the last word dropped.

Now assume we have a mapping $\mathcal{C}$ from our target vocabulary V into a set of classes K, with $|K| \ll |V|$

$$\begin{aligned} \mathcal{C} : V &\to K \\ w &\mapsto \mathcal{C}_w \end{aligned} \quad (3)$$

We can extend the mapping to a sequence of words w_1^N just by concatenating the mappings of the individual words, i.e. $\mathcal{C}_{w_1^N} = \mathcal{C}_{w_1} \dots \mathcal{C}_{w_N}$.

Given this mapping we now define our heuristic by taking the maximum LM probability associated with the words that get mapped to the same class. More formally, define the following functions corresponding to the quantities α and γ of equation 2

$$\alpha_{\mathcal{H}}(w|h) = \max_{\substack{w':\mathcal{C}_{w'}=\mathcal{C}_w \\ h':\mathcal{C}_{h'}=\mathcal{C}_h}} \left\{\alpha(w'|h')\right\} \quad (4)$$

$$\gamma_{\mathcal{H}}(h) = \max_{h':\mathcal{C}_{h'}=\mathcal{C}_h} \left\{\gamma(h')\right\} \quad (5)$$

and the resulting heuristic

$$\mathcal{H}(w|h) = \begin{cases} \alpha_{\mathcal{H}}(w|h) & \text{if } N(\mathcal{C}_w|\mathcal{C}_h) > 0 \\ \gamma_{\mathcal{H}}(h)\alpha_{\mathcal{H}}(w|\overline{h}) & \text{if } N(\mathcal{C}_w|\mathcal{C}_h) = 0 \end{cases} \quad (6)$$

The parameters of this heuristic function can be computed offline before the actual translation process and are stored in ARPA-format, like a normal LM. This allows the reuse of the existing code for handling language models.

Note that $\mathcal{H}(w|h)$ does not define a probability distribution any more, as it is not normalized. This poses no problem, as we are looking for an upper bound of the language model probabilities, and these do not need to form a probability distribution themselves.

This heuristic value is computed for the derivations as they are being produced, and it gets updated in the corresponding hyperedge. The motivation for this heuristic is that the expected similarity of the words which can be produced by the translation rules associated with an hyperedge and the contexts in this hyperedge can be captured with the given classes, and thus this optimistic language model score is able to predict future LM scores.

One could also think of a, at least at first glance, more straightforward approach. Given the mapping of words into classes, we could compute the mapping of the data used for training the language model, and then train a new language model on this data. This approach, however, has a big drawback for the usage as an heuristic. If a new language model is trained, the probabilities associated with it are in a completely different range, due to the reduced vocabulary size. Therefore the newly trained language model does not give enough information about the original language model.

5.1 Acceptability

Is this heuristic optimistic (and thus acceptable)? Taking into account the derivations for which we compute the heuristic, in most of the cases it is. This is because we take the maximum of every term involved in Equation 2. Note however, that the conditions in the case distinction have changed. In particular we move from testing the presence of a word-history pair to the presence of the corresponding classes. As the classes are more general than the words it can be the case that for some combination we use the event-seen case (first line in the case distinction of Equations 2 and 6) instead of the backoff case used when considering the words themselves. In practice, the probability of the event-seen case is expected to be higher, but we can not guarantee it.

Another source of discrepancy arises from the term $\gamma(h)$ (and the corresponding $\gamma_{\mathcal{H}}(h)$) and unseen histories h. Again, it can happen that in considering $\mathcal{C}_h$ we shift from an unseen to a seen event. Depending on the definition of the γ function this can have issues on the acceptability of the heuristic

function. In our concrete case, we train our models using Kneser-Ney smoothing (Kneser and Ney, 1995) and use the SRI toolkit (Stolcke, 2002) for our implementation. Under this conditions, for unseen histories, $\gamma(h) = 1$ (or gets a cost of 0, in the negative log-probability space). That means that when C_h has been seen, our heuristic will again not be acceptable. This, however, does not seem to have a big negative effect on the results.

The generalization on other hypotheses along the same hyperedge cannot be guaranteed, but experiments on this respect are presented on Section 7.

With respect to efficiency, this heuristic introduces a new language model into the translation process. However, the size of this language model is quite small, especially when compared with the full language model used in search, and thus the overhead of the additional LM computations is small. On the other side, when compared with the original heuristic, we eliminate the need of the -LM pass altogether.

5.2 Choosing the Classes

There is still the open question of how to choose the word-to-class mapping $\mathcal{C}$. In our case we investigated two alternatives. The first one is to use automatically generated classes. We used the `mkcls` tool (Och, 1999), which uses a maximum likelihood approach on a corpus by using a class bigram decomposition. This tool is widely used as part of the preprocessing steps when training statistical alignments using the GIZA++ tool (Och and Ney, 2003). This criterion seems to be adequate for our task, as both the words themselves and the context are taken into account.

Another possibility would be to use Part-of-Speech tags as word classes. The tagging itself can, however, be an expensive process, involving a new search in itself. We applied a simplifying assumption, in which we remove the ambiguity of the tagging. We applied a full POS-tagger (Brants, 2000) to the training corpora and then we simply selected the most frequent POS tag for each word. In this way we defined our mapping $\mathcal{C}$.

6 Experimental Results

Experiments are reported using the 2008 WMT evaluation data (Callison-Burch et al., 2008), for the German-to-English translation direction. This corpus consists of the speeches held in the plenary session of the European Parliament. The test data was the in-domain data used in the evaluation. The statistics of the corpus can be seen in Table 1.

Figure 2 shows the results for the -LM heuristic[3]. The BLEU score is shown in Figure 2(a). The best results are achieved with a -LM n-best size of 200. The difference in performance, however is not too big and nearly optimal results can already be achieved with a -LM n-best size of 50. When looking into the computational resources the difference, however, becomes critical. Figure 2(b) shows the memory usage dependent on the -LM n-best size. We can see that the memory requirements grow nearly linearly with the size of the n-best list (which is to be expected). The memory requirements using a -LM 50-best list is around 1.6GB. When using the 200-best list for optimal performance the memory requirements grow up to 6.5GB. For n-best sizes greater then 400, the memory requirements become prohibitive for the majority of current computers.

Computation time requirements are shown in Figure 2(c), as the average time needed for translating a sentence. The time requirements also grow with increasing -LM n-best size, but they stay quite reasonable, with a maximum of 6.5s per sentence. For optimum performance (200-best list), 5.2s per sentence are needed, for a 50-best heuristic, 4.3s. All time measures were taken on machines equipped with Quad-Core AMD Opteron processors, with a clock speed of 2.2GHz.

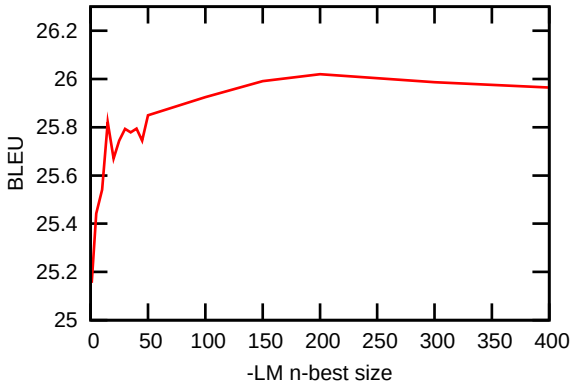
The results for the coarse LM heuristic are show in Figure 3. It can be seen that the performance of the system using this heuristic is comparable or even slightly superior in the best case, however only marginally so. The behaviour of this heuristic is somewhat more erratic than in -LM case. Memory requirements are shown in Figure 3(b). The memory requirements using the coarse LM heuristic are much lower than when using the -LM heuristic (note the different scale on the y-axis between Figures 2(b) and 3(b)), and they get less as the number of classes increases.

Time requirements are shown in Figure 3(c) and are in general lower than for the case of -LM heuristics, except for very small values of n, where the translation performance suffers severely. The time requirements also show an erratic behaviour. However, different workloads of the ma-

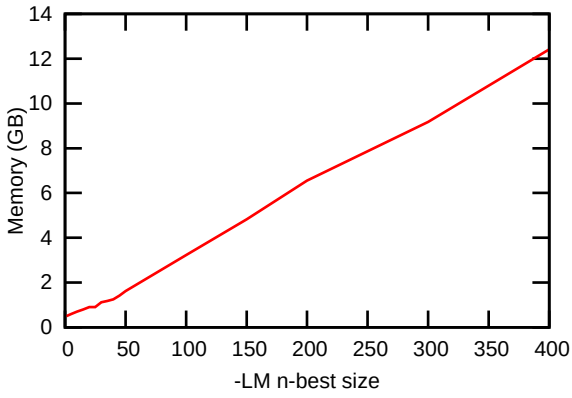
[3]Note that in our implementation we used hypothesis recombination also in the -LM pass

	Training set		Test set	
	German	English	German	English
Sentences	1,266,520		2,000	
Words	33 404 503	35 259 758	56 624	60 185
Distinct words	301 006	96 802	8 844	6 050

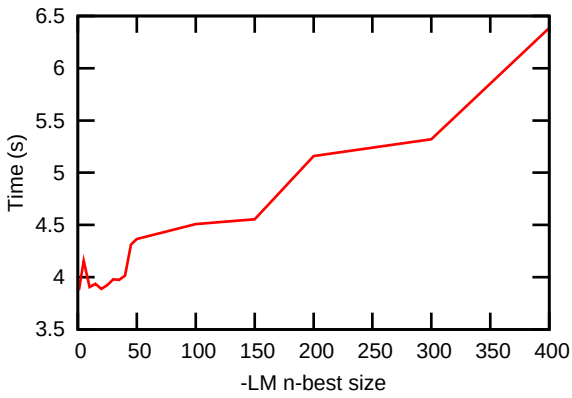
Table 1: Corpora statistics



(a) BLEU score

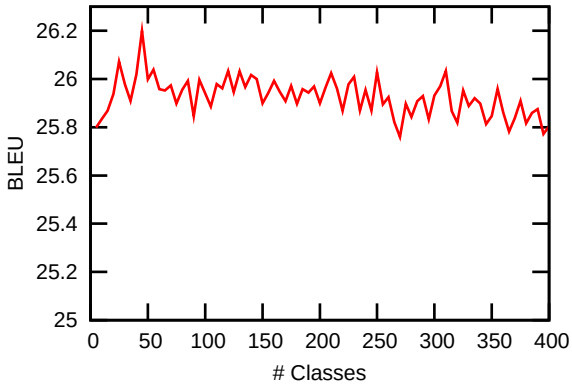


(b) Memory

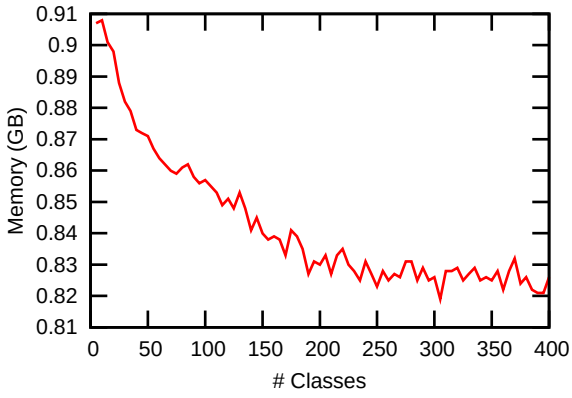


(c) Time per sentence

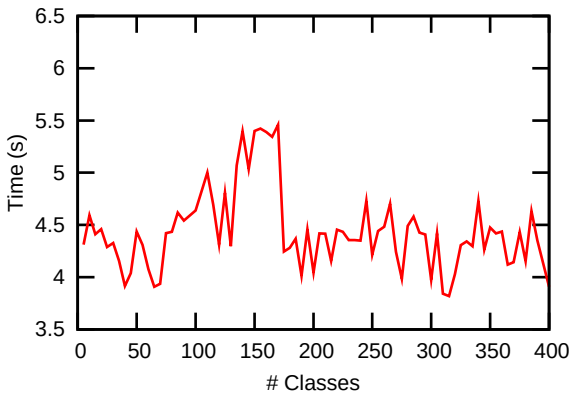
Figure 2: Results using the -LM heuristic



(a) BLEU score



(b) Memory



(c) Time per sentence

Figure 3: Results using the coarse LM heuristic

chines at experimentation time probably had a non-negligible effect on these measurements.

Using POS as classes does not seem to improve performance. It achieves a BLEU score of 25.9%, and the memory and time requirements are comparable with those of the equivalent number of automatic classes (we work with 41 POS classes).

7 Discussion

The behaviour of the -LM heuristic was expected. The increase in memory and time requirements is due to the increase effort in generating the -LM n-best lists. This does not imply an increase in translation quality, as, probably, the new hyperedges that get considered in the heuristic computation do not get used in the actual translation process.

Figure 4 shows how many times the -LM heuristic was not acceptable, computed for the first 100 sentences of the test corpus. As expected, this number decreases as the size of the n-best list increases, but starting from a value of around 100 the rate of decrease of failed heuristics is much lower. This explains the behaviour of the BLEU score shown in Figure 2(a).

The coarse-LM heuristic already achieves a good performance even for a small number of classes. This heuristic is able to simplify the LM computation scores and guide the parsing process in an efficient manner. This is consistent with the findings of (Petrov et al., 2008), albeit in a slightly different context (Petrov et al. used the coarse LM for pruning purposes).

This observation can be confirmed in Figure 5, where the coarse heuristic produces much less heuristic failures as the -LM heuristic. Somewhat counter-intuitively, however, the number of LM heuristic fails increases with the number of classes. This can be explained by the fact, that, in spite of the chances of incurring into one of the failure cases exposed in Section 5.1 grow when considering a small amount of classes, the maximum probability induced by the mapping C also is greater as the number of classes diminishes. This can be clearly seen by considering only one class. In spite of certainly incurring in one of the "fail" cases, we certainly will get an optimistic estimation of the LM score. In this way, a small number of classes does not provide so good information to accurately discriminate between the candidate derivations, which explains the higher cost with a small number of classes. In this case, the absolute bound on the number of candidates pointed to at the end of Section 3 has a higher probability of coming into effect.

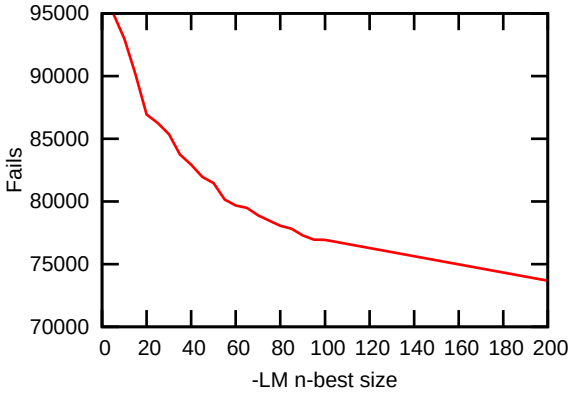


Figure 4: -LM heuristic fails.

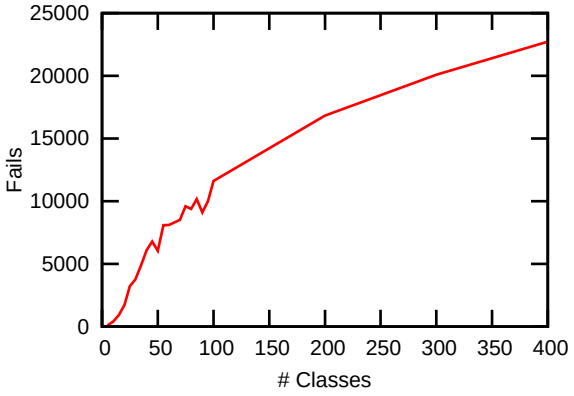


Figure 5: Coarse LM heuristic fails.

An increase in the number of classes reduces the memory and time requirements, which is an indication that the search effort gets more focused. This is not reflected in an increase in BLEU score. The behaviour of the BLEU curve is however somewhat erratic and, nevertheless stays on a range of 0.2%, so no clear conclusion can be drawn from that. Of course, an increase in the number of classes also implies an increase in the lookup time (but with sizes up to 400, as we have done in this paper, this is negligible). This also involves an increase in (offline) computation time for clustering the words. In our experiments we selected the maximum number of classes that we could compute in 24 hours[4].

[4]However on some less powerful computers as the ones the translation experiments were carried on

8 Conclusions

In this paper we have studied the language model heuristic proposed by (Huang and Chiang, 2007) where they describe the cube growing algorithm. We have analysed the performance and efficiency when varying the size of the n-best list required for the heuristic computation. We have proposed a new heuristic, based on taking the maximum of the LM scores which achieves the same (or marginally better) performance but using significant less memory and with improvements in running time.

We just tried two approaches to word clustering, the automatic one implemented by the well-known tool `mkcls` and the one based on POS classes. Although no big difference in performance or efficiency could be found between these two word clusterings, perhaps a smarter and more task-directed clustering criterion can further improve the results. Specially, a bilingual word clustering algorithm, where both the source and target language words are taken into account, will probably provide better performance, as the similarity between the words that may occur as translations along an hyperedge may be better modelled.

Acknowledgements

This work was realised as part of the Quaero Programme, funded by OSEO, French State agency for innovation.

This material is based upon work supported by the Defense Advanced Research Projects Agency (DARPA) under Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the Defense Advanced Research Projects Agency (DARPA).

References

Brants, Thorsten. 2000. Tnt – a statistical part-of-speech tagger. In *Proceedings of the 6th Applied Natural Language Processing Conference (ANLP)*, pages 224–231, Seattle, WA.

Callison-Burch, Chris, Cameron Fordyce, Philipp Koehn, Christof Monz, and Josh Schroeder. 2008. Further meta-evaluation of machine translation. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 70–106, Columbus, Ohio, June. Association for Computational Linguistics.

Chappelier, JC and M. Rajman. 1998. A generalized CYK algorithm for parsing stochastic CFG. In *First Workshop on Tabulation in Parsing and Deduction (TAPD98)*, pages 133–137.

Chiang, D. 2005. A Hierarchical Phrase-Based Model for Statistical Machine Translation. *Ann Arbor*, 100.

Chiang, D. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33(2):201–228.

Huang, L. and D. Chiang. 2007. Forest Rescoring: Faster Decoding with Integrated Language Models. In *ANNUAL MEETING-ASSOCIATION FOR COMPUTATIONAL LINGUISTICS*, volume 45, page 144.

Kneser, R. and H. Ney. 1995. Improved backing-off for M-gram language modeling. In *Acoustics, Speech, and Signal Processing, 1995. ICASSP-95., 1995 International Conference on*, volume 1.

Koehn, P. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. *Washington DC*.

Marcu, D., W. Wang, A. Echihabi, and K. Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. *Proceedings of EMNLP*, pages 44–52.

Och, Franz Josef and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

Och, F.J. 1999. An efficient method for determining bilingual word classes. In *EACL99: Ninth Conf. of the Europ. Chapter of the Association for Computational Linguistics*, pages 71–76.

Petrov, Slav, Aria Haghighi, and Dan Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing*, pages 108–116, Honolulu, Hawaii, October. Association for Computational Linguistics.

Stolcke, A. 2002. SRILM-an Extensible Language Modeling Toolkit. In *Seventh International Conference on Spoken Language Processing*. ISCA.