Computing Optimal Alignments for the IBM-3 Translation Model

Thomas Schoenemann
Centre for Mathematical Sciences
Lund University, Sweden

Abstract

Prior work on training the IBM-3 translation model is based on suboptimal methods for computing Viterbi alignments. In this paper, we present the first method guaranteed to produce globally optimal alignments. This not only results in improved alignments, it also gives us the opportunity to evaluate the quality of standard hillclimbing methods. Indeed, hillclimbing works reasonably well in practice but still fails to find the global optimum for between 2% and 12% of all sentence pairs and the probabilities can be several tens of orders of magnitude away from the Viterbi alignment.

By reformulating the alignment problem as an Integer Linear Program, we can use standard machinery from global optimization theory to compute the solutions. We use the well-known branch-and-cut method, but also show how it can be customized to the specific problem discussed in this paper. In fact, a large number of alignments can be excluded from the start without losing global optimality.

1 Introduction

Brown et al. (1993) proposed to approach the problem of automatic natural language translation from a statistical viewpoint and introduced five probability models, known as IBM 1-5. Their models were *single word based*, where each source word could produce at most one target word.

State-of-the-art statistical translation systems follow the *phrase based* approach, e.g. (Och and Ney, 2000; Marcu and Wong, 2002; Koehn, 2004; Chiang, 2007; Hoang et al., 2007), and hence allow more general alignments. Yet, single word based models (Brown et al., 1993; Brown et al., 1995; Vogel et al., 1996) are still highly relevant: many phrase based systems extract phrases from the alignments found by training the single word based models, and those that train phrases directly usually underperform these systems (DeNero et al., 2006).

Single word based models can be divided into two classes. On the one hand, models like IBM-1, IBM-2 and the HMM are computationally easy to handle: both marginals and Viterbi alignments can be computed by dynamic programming or even simpler techniques.

On the other hand there are fertility based models, including IBM 3-5 and Model 6. These models have been shown to be of higher practical relevance than the members of the first class (Och and Ney, 2003) since they usually produce better alignments. At the same time, computing Viterbi alignments for these methods has been shown to be NP-hard (Udupa and Maji, 2006), and computing marginals is no easier.

The standard way to handle these models – as implemented in GIZA++ (Al-Onaizan et al., 1999; Och and Ney, 2003) – is to use a hillclimbing algorithm. Recently Udupa and Maji (2005) proposed an interesting approximation based on solving sequences of exponentially large subproblems by means of dynamic programming and also addressed the decoding problem. In both cases there is no way to tell how far away the result is from the Viterbi alignment.

In this paper we solve the problem of finding IBM-3 Viterbi alignments by means of Integer Linear Programming (Schrijver, 1986). While there is no polynomial run-time guarantee, in practice the applied branch-and-cut framework is fast enough to find optimal solutions even for the large Canadian Hansards task (restricted to sentences with at most 75 words), with a training time of 6 hours on a 2.4 GHz Core 2 Duo (single threaded).

Proceedings of the Fourteenth Conference on Computational Natural Language Learning, pages 98–106,
Uppsala, Sweden, 15-16 July 2010.

Integer Linear Programming in the context of machine translation first appeared in the work of Germann et al. (2004), who addressed the *translation* problem (often called *decoding*) in terms of a travelings-salesman like formulation. Recently, DeNero and Klein (2008) addressed the training problem for phrase-based models by means of integer linear programming, and proved that the problem is NP-hard. The main difference to our work is that they allow only consecutive words in the phrases. In their formulation, allowing arbitrary phrases would require an exponential number of variables. In contrast, our approach handles the classical single word based model where any kind of "phrases" in the source sentence are aligned to one-word phrases in the target sentence.

Lacoste-Julien et al. (2006) propose an integer linear program for a symmetrized word-level alignment model. Their approach also allows to take the alignments of neighboring words into account. In contrast to our work, they only have a very crude fertility model and they are considering a substantially different model. It should be noted, however, that a subclass of their problems can be solved in polynomial time - the problem is closely related to bipartite graph matching. Less general approaches based on matching have been proposed in (Matusov et al., 2004) and (Taskar et al., 2005).

Recently Bodrumlu et al. (2009) proposed a very innovative cost function for jointly optimizing dictionary entries and alignments, which they minimize using integer linear programming. They also include a mechanism to derive N-best lists. However, they mention rather long computation times for rather small corpora. It is not clear if the large Hansards tasks could be addressed by their method.

An overview of integer linear programming approaches for natural language processing can be found on `http://ilpnlp.wikidot.com/`. To facilitate further research in this area, the source code will be made publicly available.

Contribution The key contribution of our work is a method to handle exact fertility models as arising in the IBM-3 model in a global optimization framework. This is done by a linear number of linear consistency constraints. Unlike all previous works on integer linear programming for machine translation, we do not solely use binary coefficients in the constraint matrix, hence showing that the full potential of the method has so far not been explored.

At the same time, our method allows us to give a detailed analysis of the quality of hillclimbing approaches. Moreover, we give a more detailed description of how to obtain a fast problem-tailored integer solver than in previous publications, and include a mechanism to a priori exclude some variables without losing optimality.

2 The IBM-3 Translation Model

Given a source sentence f_1^J, the statistical approach to machine translation is to assign each possible target sentence e_1^I a probability to be an accurate translation. For convenience in the translation process, this probability is usually rewritten as

$$P(e_1^I|f_1^J) = \frac{1}{p(f_1^J)} \cdot p(e_1^I) \cdot p(f_1^J|e_1^I) \ ,$$

and the training problem is to derive suitable parameters for the latter term from a bilingual corpus. Here, the probability is expressed by summing over hidden variables called *alignments*. The common assumption in single word based models is that each source position j produces a single target position $a_j \in \{0, \ldots, I\}$, where an artificial 0-position has been introduced to mark words without a correspondence in the target sentence. The alignment of a source sentence is then a vector a_1^J, and the probability can now be written as

$$p(f_1^J|e_1^I) = \sum_{a_1^J} p(f_1^J, a_1^J|e_1^I) \ .$$

We will focus on training the IBM-3 model which is based on the concept of *fertilities*: given an alignment a_1^J, the fertility $\Phi_i(a_1^J) = \sum_{j:a_j=i} 1$ of target word i expresses the number of source words aligned to it. Omitting the dependence on a_1^J (and defining $p(j|0) = 1$), the probability is expressed as

$$\begin{aligned} p(f_1^J, a_1^J|e_1^I) &= p(\Phi_0|J) \cdot \prod_{i=1}^{I} \Big[\Phi_i! \, p(\Phi_i|e_i)\Big] \\ &\quad \cdot \prod_j \Big[p(f_j|e_{a_j}) \cdot p(j|a_j)\Big] \ . \end{aligned} \tag{1}$$

For the probability $p(\Phi_0|J)$ of the fertility of the empty word, we use the modification introduced in (Och and Ney, 2003), see there for details. In summary, the model comprises a single word based

translation model, an inverted zero-order alignment model and a fertility model. We now discuss how to find the optimal alignment for given probabilities, i.e. to solve the problem

$$\arg\max_{a_1^J} \; p(f_1^J, a_1^J | e_1^I) \qquad (2)$$

for each bilingual sentence pair in the training set. This is a desirable step in the approximate EM-algorithm that is commonly used to train the model.

3 Finding IBM-3 Viterbi Alignments via Integer Linear Programming

Instead of solving (2) directly we consider the equivalent task of minimizing the negative logarithm of the probability function. A significant part of the arising cost function is already linear in terms of the alignment variables, a first step for the integer linear program (ILP) we will derive.

To model the problem as an ILP, we introduce two sets of variables. Firstly, for any source position $j \in \{1, \ldots, J\}$ and any target position $i \in \{0, \ldots, I\}$ we introduce an integer variable $x_{ij} \in \{0, 1\}$ which we want to be 1 exactly if $a_j = i$ and 0 otherwise. Since each source position must be aligned to exactly one target position, we arrive at the set of linear constraints

$$\sum_i x_{ij} = 1 \quad , \; j = 1, \ldots, J \; . \qquad (3)$$

The negative logarithm of the bottom row of (1) is now easily written as a linear function in terms of the variables x_{ij}:

$$\sum_{i,j} c_{ij}^x \cdot x_{ij} \quad ,$$

$$c_{ij}^x = -\log \Big[p(f_j | e_i) \cdot p(j|i) \Big] \; .$$

For the part of the cost depending on the fertilities, we introduce another set of integer variables $y_{if} \in \{0, 1\}$. Here $i \in \{0, \ldots, I\}$ and f ranges from 0 to some pre-specified limit on the maximal fertility, which we set to $\max(15, J/2)$ in our experiments (fertilities $> J$ need not be considered). We want y_{if} to be 1 if the fertility of i is f, 0 otherwise. Hence, again these variables must sum to 1:

$$\sum_f y_{if} = 1 \quad , \; i = 0, \ldots, I \; . \qquad (4)$$

The associated part of the cost function is written as

$$\sum_{i,f} c_{if}^y \cdot y_{if} \quad ,$$

$$c_{if}^y = -\log \Big[f! \, p(f | e_i) \Big] \quad , \; i = 1, \ldots, I$$

$$c_{0f}^y = -\log \Big[p(\Phi_0 = f | J) \Big] \; .$$

It remains to ensure that the variables y_{if} expressing the fertilities are consistent with the fertilities induced by the alignment variables x_{ij}. This is done via the following set of linear constraints:

$$\sum_j x_{ij} = \sum_f f \cdot y_{if} \quad , \; i = 0, \ldots, I \; . \qquad (5)$$

Problem (2) is now reduced to solving the integer linear program

$$\arg\min_{\{x_{ij}\}, \{y_{if}\}} \; \sum_{i,j} c_{ij}^x \, x_{ij} + \sum_{i,f} c_{if}^y \, y_{if}$$

$$\text{subject to } (3), \; (4), \; (5)$$

$$x_{ij} \in \{0, 1\}, \;\; y_{if} \in \{0, 1\} \; , \qquad (6)$$

with roughly $2\,I\,J$ variables and roughly $J + 2I$ constraints.

4 Solving the Integer Linear Program

To solve the arising integer linear programming problem, we first relax the integrality constraints on the variables to continuous ones:

$$x_{ij} \in [0, 1], \;\; y_{if} \in [0, 1] \; ,$$

and obtain a lower bound on the problems by solving the arising linear programming relaxation via the dual simplex method.

While in practice this can be done in a matter of milli-seconds even for sentences with $I, J > 50$, the result is frequently a fractional solution. Here the alignment variables are usually integral but the fertility variables are not.

In case the LP-relaxation does not produce an integer solution, the found solution is used as the initialization of a branch-and-cut framework. Here one first tries to strengthen the LP-relaxation by deriving additional inequalities that must be valid for all integral solutions see e.g. (Schrijver, 1986; Wolter, 2006) and `www.coin-or.org`. These inequalities are commonly called *cuts*. Then one applies a branch-and-bound scheme on the integer variables. In each step of this scheme, additional inequalities are derived. The process is further sped-up by introducing a heuristic to derive an

upper bound on the cost function. Such bounds are generally given by feasible integral solutions. We use our own heuristic as a plug-in to the solver. It generates solutions by thresholding the alignment variables (winner-take-all) and deriving the induced fertility variables. An initial upper bound is furthermore given by the alignment found by hillclimbing.

We suspect that further speed-ups are possible by using so-called follow-up nodes: e.g. if in the branch-and-bound an alignment variable x_{ij} is set to 1, one can conclude that the fertility variable y_{i0} must be 0. Also, sets of binary variables that must sum to 1 as in (3) and (4) are known as *special ordered sets of type I* and there are variants of branch-and-cut that can exploit these properties. However, in our context they did not result in speed-ups.

Our code is currently based on the open source COIN-OR project[1] and involves the linear programming solver CLP, the integer programming solver CBC, and the cut generator library CGL. We have also tested two commercial solvers. For the problem described in this paper, CBC performed best. Tests on other integer programming tasks showed however that the Gurobi solver outperforms CBC on quite a number of problems.

5 Speed-ups by Deriving Bounds

It turns out that, depending on the cost function, some variables may a priori be excluded from the optimization problem without losing global optimality. That is, they can be excluded even *before* the first LP-relaxation is solved.

The affected variables have relatively high cost coefficients and they are identified by considering lower bounds and an upper bound on the cost function. Starting from the lower bounds, one can then identify variables that when included in a solution would raise the cost beyond the upper bound.

An upper bound u on the problem is given by any alignment. We use the one found by hillclimbing. If during the branch-and-cut process tighter upper bounds become available, the process could be reapplied (as a so-called *column cut generator*).

For the lower bounds we use different ones to exclude alignment variables and to exclude fertility variables.

[1] `www.coin-or.org`

5.1 Excluding Alignment Variables

To derive a lower bound for the alignment variables, we first observe that the cost c^x_{ij} for the alignment variables are all positive, whereas the cost c^y_{if} for the fertilities are frequently negative, due to the factorial of f. A rather tight lower bound on the fertility cost can be derived by solving the problem

$$\begin{aligned} l_{F,1} = \quad & \min_{\{\Phi_i\}} \sum_{i=0}^{I} c^y_{i\Phi_i} \\ & \text{s.t. } \textstyle\sum_i \Phi_i = J \quad , \end{aligned} \tag{7}$$

which is easily solved by dynamic programming proceeding along i. A lower bound on the alignment cost is given by

$$l_A = \textstyle\sum_j l_{A,j} \ ,$$
$$\text{where} \quad l_{A,j} = \min_{i=0,\ldots,I} c^x_{ij} \ .$$

The lower bound is then given by $l_1 = l_{F,1} + l_A$, and we can be certain that source word j will not be aligned to target word i if

$$c^x_{ij} > l_{A,j} + (u - l_1) \ .$$

5.2 Excluding Fertility Variables

Excluding fertility variables is more difficult as cost can be negative and we have used a constraint to derive $l_{F,1}$ above.

At present we are using a two ways to generate a lower bound and apply the exclusion process with each of them sequentially. Both bounds are looser than l_1, but they immensely help to get the computation times to an acceptable level.

The first bound builds upon l_1 as derived above, but using a looser bound $l_{F,2}$ for the fertility cost:

$$l_{F,2} = \sum_i \min_{\Phi_i} c^y_{i\Phi_i} \ .$$

This results in a bound $l_2 = l_{F,2} + l_A$, and fertility variables can now be excluded in a similar manner as above.

Our second bound is usually much tighter and purely based on the fertility variables:

$$l_3 = \sum_i \min_{\Phi_i} \left[c^y_{i\Phi_i} + \min_{\mathcal{J} \subseteq \{1,\ldots,J\} : |\mathcal{J}| = \Phi_i} c^x_i(\mathcal{J}) \right] ,$$

$$\text{with } c^x_i(\mathcal{J}) = \sum_{j \in \mathcal{J}} c^x_{ij} \ ,$$

and where the cost of the empty set is defined as 0. Although this expression looks rather involved, it is actually quite easy to compute by simply sorting the respective cost entries. A fertility variable y_{if} can now be excluded if the difference between c^y_{if} and the contribution of i to l_3 exceeds $u - l_3$.

We consider it likely that more variables can be excluded by deriving bounds in the spirit of (7), but with the additional constraint that $\Phi_i = f$ for some i and f. We leave this for future work.

6 Experiments

We have tested our method on three different tasks involving a total of three different languages and each in both directions. The first task is the well-known Canadian Hansards[2] task (senate debates) for French and English. Because of the large dataset we are currently only considering sentence pairs where both sentences have at most 75 words. Longer sentences are usually not useful to derive model parameters.

The other two datasets are released by the European Corpus Initiative[3]. We choose the Union Bank of Switzerland (UBS) corpus for English and German and the Avalanche Bulletins, originally released by SFISAR, for French and German. For the latter task we have annotated alignments for 150 of the training sentences, where one annotator specified sure and possible alignments. For details, also on the alignment error rate, see (Och and Ney, 2003).

All corpora have been preprocessed with language-specific rules; their statistics are given in Table 1. We have integrated our method into the standard toolkit GIZA++[4] and are using the training scheme $1^5H^53^54^5$ for all tasks. While we focus on the IBM-3 stage, we also discuss the quality of the resulting IBM-4 parameters and alignments.

Experiments were run on a 2.4 GHz Core 2 Duo with 4 GB memory. For most sentence pairs, the memory consumption of our method is only marginally more than in standard GIZA++ (600 MB). In the first iteration on the large Hansards task, however, there are a few very difficult sentence pairs where the solver needs up to 90 minutes and 1.5 GB . We observed this in both translation directions.

[2] `www.isi.edu/natural-language/download/hansard/`

[3] The entire CD with many more corpora is available for currently 50 Euros.

[4] available at `code.google.com/p/giza-pp/`.

Avalanche Bulletin		
	French	German
# sentences	2989	
max. sentence length	88	57
total words	64825	45629
vocabulary size	1707	2113

UBS		
	English	German
# sentences	2689	
max. sentence length	92	91
total words	62617	53417
vocabulary size	5785	9127

Canadian Hansards (max. 75)		
	French	English
# sentences	180706	
max. sentence length	75	75
total words	3730570	3329022
vocabulary size	48065	37633

Table 1: **Corpus statistics** for all employed (training) corpora, after preprocessing.

6.1 Evaluating Hillclimbing

In our first set of experiments, we compute Viterbi alignments merely to evaluate the quality of the standard training process. That is, the model parameters are updated based on the alignments found by hillclimbing. Table 2 reveals that, as expected, hillclimbing does not always find the global optimum: depending on the task and iteration number, between 2 and 12 percent of all hillclimbing alignments are suboptimal. For short sentences (i.e. $I, J \leq 20$) hillclimbing usually finds the global optimum.

Somewhat more surprisingly, even when a good and hence quite focused initialization of the IBM-3 model parameters is given (by training HMMs first), the probability of the Viterbi alignment can be up to a factor of 10^{37} away from the optimum. This factor occurred on the Hansards task for a sentence pair with 46 source and 46 target words and the fertility of the empty word changed from 9 (for hillclimbing) to 5.

6.2 Hillclimbing vs. Viterbi Alignments

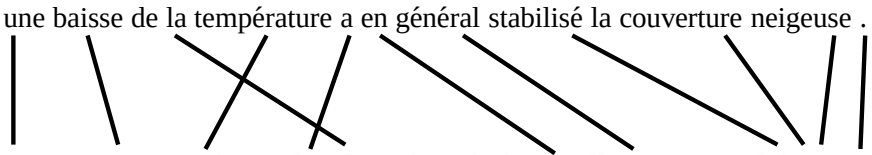
We now turn to a training scheme where the Viterbi alignments are used to actually update the model parameters, and compare it to the standard training scheme (based on hillclimbing).

	Candian Hansards (max 75)				
	French → English				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	10.7%	10.7%	10.8%	11.1%	11.4%
Maximal factor to Viterbi alignment	$1.9 \cdot 10^{37}$	$9.1 \cdot 10^{17}$	$7.3 \cdot 10^{14}$	$3.3 \cdot 10^{12}$	$8.1 \cdot 10^{14}$
	English → French				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	7.3%	7.5%	7.4%	7.4%	7.5%
Maximal factor to Viterbi alignment	$5.6 \cdot 10^{38}$	$6.6 \cdot 10^{20}$	$7.6 \cdot 10^{11}$	$4.3 \cdot 10^{10}$	$8.3 \cdot 10^{11}$

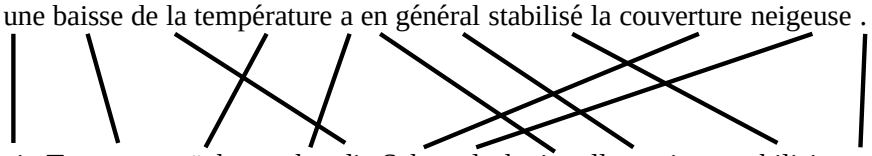
	Avalanche Bulletins				
	French → German				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	7.5%	5.6%	4.9%	4.9%	4.4%
Maximal factor to Viterbi alignment	$6.1 \cdot 10^{5}$	877	368	$2.5 \cdot 10^{4}$	429
	German → French				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	4.2%	2.7%	2.5%	2.3%	2.1%
Maximal factor to Viterbi alignment	40	302	44	$3.3 \cdot 10^{4}$	$9.2 \cdot 10^{4}$

	Union Bank of Switzerland (UBS)				
	English → German				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	5.0%	4.0%	3.5%	3.3%	3.2%
Maximal factor to Viterbi alignment	677	22	53	40	32
	German → English				
Iteration #	1	2	3	4	5
# suboptimal alignments in hillclimbing	5.5%	3.3%	2.5%	2.2%	2.3%
Maximal factor to Viterbi alignment	$1.4 \cdot 10^{7}$	808	33	33	$1.8 \cdot 10^{4}$

Table 2: **Analysis of Hillclimbing on all considered tasks.** All numbers are for the IBM-3 translation model. Iteration 1 is the first iteration after the transfer from HMM, the final iteration is the transfer to IBM4. The factors are w.r.t. the original formulation, not the negative logarithm of it and are defined as the maximal ratio between the Viterbi probability and the hillclimbing probability.



Standard training (hillclimbing).



Proposed training (Viterbi alignments).

Figure 1: **Comparison of training schemes.** Shown are the alignments of the final IBM-3 iteration.

Indeed Table 3 demonstrates that with the new training scheme, the perplexities of the final IBM-3 iteration are consistently lower. Yet, this effect does not carry over to IBM-4 training, where the perplexities are consistently higher. Either this is due to overfitting or it is better to use the same method for alignment computation for both IBM-3 and IBM-4. After all, both start from the HMM Viterbi alignments.

Interestingly, the maximal factor between the hillclimbing alignment and the Viterbi alignment is now consistently higher on all tasks and in all iterations. The extreme cases are a factor of 10^{76} for the Canadian Hansards English $\rightarrow$ French task and 10^{30} for the Bulletin French $\rightarrow$ German task.

Table 4 demonstrates that the alignment error rates of both schemes are comparable. Indeed, a manual evaluation of the alignments showed that most of the changes affect words like articles or prepositions that are generally hard to translate. In many cases neither the heuristic nor the Viterbi alignment could be considered correct. An interesting case where the proposed scheme produced the better alignment is shown in Figure 1.

In summary, our results give a thorough justification for the commonly used heuristics. A test with the original non-deficient empty word model of the IBM-3 furthermore confirmed the impression of (Och and Ney, 2003) that overly many words are aligned to the empty word: the tendency is even stronger in the Viterbi alignments.

6.3 Optimizing Running Time

The possibilities to influence the run-times of the branch-and-cut framework are vast: there are nu-

	Union Bank (UBS) E $\rightarrow$ G	
	Final IBM-3	Final IBM-4
Standard train.	49.21	35.73
Proposed train.	49.00	35.76

	Union Bank (UBS) G $\rightarrow$ E	
	Final IBM-3	Final IBM-4
Standard train.	62.38	47.39
Proposed train.	62.08	47.43

	Avalanche F $\rightarrow$ G	
	Final IBM-3	Final IBM-4
Standard train.	35.44	21.99
Proposed train.	35.23	22.04

	Avalanche G $\rightarrow$ F	
	Final IBM-3	Final IBM-4
Standard train.	34.60	22.78
Proposed train.	34.48	22.76

	Canadian Hansards F $\rightarrow$ E	
	Final IBM-3	Final IBM-4
Standard train.	105.28	55.22
Proposed train.	92.09	55.35

	Canadian Hansards E $\rightarrow$ F	
	Final IBM-3	Final IBM-4
Standard train.	70.58	37.64
Proposed train.	70.03	37.73

Table 3: **Analysis of the perplexities in training.**

	French → German	
	Final IBM-3	Final IBM-4
Standard train.	24.31%	23.01%
Proposed train.	24.31%	23.24%

	German → French	
	Final IBM-3	Final IBM-4
Standard train.	33.03%	33.44%
Proposed train.	33.00%	33.27%

Table 4: **Alignment error rates** on the Avalanche bulletin task.

merous ways to generate cuts and several of them can be used simultaneously. The CBC-package also allows to specify how many rounds of cuts to derive at each node. Then there is the question of whether to use the bounds derived in Section 5 to a priori exclude variables. Finally, branch-and-cut need not be done on all variables: since solving LP-relaxations typically results in integral alignments, it suffices to do branch-and-cut on the fertility variables and only add the alignment variables in case non-integral values arise (this never happened in our experiments[5]).

We could not possibly test all combinations of the listed possibilities, and our primary focus was to achieve acceptable run-times for the large Hansards task. Still, in the end we have a quite uniform picture: the lowest run-times are achieved by using Gomory Cuts only. Moreover, including all variables for branching was between 1.5 and 2 times faster than only including fertility variables. Only by exploiting the bounds derived in Section 5 the run-times for the Hansards task in direction from English to French became acceptable. We believe that further speed-ups are possible by deriving tighter bounds, and are planning to investigate this in the future.

We end up with roughly 6 hours for the Hansards task, roughly 3 minutes for the UBS task, and about 2.5 minutes for the Avalanche task. In all cases the run-times are much higher than in the standard GIZA++ training. However, we are now getting optimality guarantees where previously one could not even tell how far away one is from the optimum. And the Viterbi alignments of several sentence pairs can of course be computed in parallel.

Lastly, we mention the possibility of setting a limit on the branch-and-cut process, either on the running time or on the number of nodes. There is then no longer a guarantee for global optimality, but at least one is getting a bound on the gap to the optimum and one can be certain that the training time will be sufficiently low.

[5] In fact, when fixing the fertility variables, the problem reduces to the polynomial time solvable assignment problem.

7 Conclusion

We present the first method to compute IBM-3 Viterbi alignments with a guarantee of optimality. In contrast to other works on integer linear programming for machine translation, our formulation is able to include a precise and very general fertility model. The resulting integer linear program can be solved sufficiently fast in practice, and we have given many comments on how problem-specific knowledge can be incorporated into standard solvers.

The proposed method allows for the first time to analyze the quality of hillclimbing approaches for IBM-3 training. It was shown that they can be very far from the optimum. At the same time, this seems to happen mostly for difficult sentences that are not suitable to derive good model parameters.

In future work we want to derive tighter bounds to a priori exclude variables, combine the method with the N-best list generation of (Bodrumlu et al., 2009) and evaluate on a larger set of corpora. Finally we are planning to test other integer programming solvers.

Acknowledgments We thank Fredrik Kahl for helpful comments and an anonymous reviewer for pointing out freely available software packages. This research was funded by the European Research Council (GlobalVision grant no. 209480).

References

Y. Al-Onaizan, J. Curin, M. Jahr, K. Knight, J. Lafferty, I. D. Melamed, F. J. Och, D. Purdy, N. A. Smith, and D. Yarowsky. 1999. Statistical machine translation, Final report, JHU workshop. `http://www.clsp.jhu.edu/ws99/`.

Tugba Bodrumlu, Kevin Knight, and Sujith Ravi. 2009. A new objective function for word alignment. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing (ILP)*, Boulder, Colorado, June.

P.F. Brown, S.A. Della Pietra, V.J. Della Pietra, and R.L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, June.

P.F. Brown, J. Cocke, S.A. Della Pietra, V.J. Della Pietra, F. Jelinek, J. Lai, and R.L. Mercer. 1995. Method and system for natural language translation. U.S. patent #5.477.451.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

J. DeNero and D. Klein. 2008. The complexity of phrase alignment problems. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, Columbus, Ohio, June.

John DeNero, Dan Gillick, James Zhang, and Dan Klein. 2006. Why generative phrase models underperform surface heuristics. In *StatMT '06: Proceedings of the Workshop on Statistical Machine Translation*, pages 31–38, Morristown, NJ, USA, June.

U. Germann, M. Jahr, K. Knight, D. Marcu, and K. Yamada. 2004. Fast decoding and optimal decoding for machine translation. *Artificial Intelligence*, 154(1–2), April.

H. Hoang, A. Birch, C. Callison-Burch, R. Zens, A. Constantin, M. Federico, N. Bertoldi, C. Dyer, B. Cowan, W. Shen, C. Moran, and O. Bojar. 2007. Moses: Open source toolkit for statistical machine translation. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 177–180, Prague, Czech Republic, June.

P. Koehn. 2004. Pharaoh: A beam search decoder for phrase-based statistical machine translation models. In *Conference of the Association for Machine Translation in the Americas (AMTA)*, pages 115–124, Washington, D.C., October.

S. Lacoste-Julien, B. Taskar, D. Klein, and M. Jordan. 2006. Word alignment via quadratic assignment. In *Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, New York, New York, June.

D. Marcu and W. Wong. 2002. A phrase-based, joint probability model for statistical machine translation. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Philadelphia, Pennsylvania, July.

E. Matusov, R. Zens, and H. Ney. 2004. Symmetric word alignments for statistical machine translation. In *International Conference on Computational Linguistics (COLING)*, Geneva, Switzerland, August.

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

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

A. Schrijver. 1986. *Theory of Linear and Integer Programming*. Wiley-Interscience Series in Discrete Mathematics and Optimization. John Wiley & Sons.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Vancouver, Canada, October.

R. Udupa and H.K. Maji. 2005. Theory of alignment generators and applications to statistical machine translation. In *The International Joint Conferences on Artificial Intelligence*, Edinburgh, Scotland, August.

R. Udupa and H.K. Maji. 2006. Computational complexity of statistical machine translation. In *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, Trento, Italy, April.

S. Vogel, H. Ney, and C. Tillmann. 1996. HMM-based word alignment in statistical translation. In *International Conference on Computational Linguistics (COLING)*, pages 836–841, Copenhagen, Denmark, August.

K. Wolter. 2006. Implementation of Cutting Plane Separators for Mixed Integer Programs. Master's thesis, Technische Universität Berlin, Germany.