Comparing Reordering Constraints for SMT Using Efficient BLEU Oracle Computation

Markus Dreyer, Keith Hall, and Sanjeev Khudanpur
Center for Language and Speech Processing
Johns Hopkins University
3400 North Charles Street, Baltimore, MD 21218 USA
{dreyer,keith_hall,khudanpur}@jhu.edu

Abstract

This paper describes a new method to compare reordering constraints for Statistical Machine Translation. We investigate the best possible (oracle) BLEU score achievable under different reordering constraints. Using dynamic programming, we efficiently find a reordering that approximates the highest attainable BLEU score given a reference and a set of reordering constraints. We present an empirical evaluation of popular reordering constraints: local constraints, the IBM constraints, and the Inversion Transduction Grammar (ITG) constraints. We present results for a German-English translation task and show that reordering under the ITG constraints can improve over the baseline by more than 7.5 BLEU points.

1 Introduction

Reordering the words and phrases of a foreign sentence to obtain the target word order is a fundamental, and potentially the hardest, problem in machine translation. The search space for all possible permutations of a sentence is factorial in the number of words/phrases; therefore a variety of models have been proposed that constrain the set of possible permutations by allowing certain reorderings while disallowing others. Some models (Brown et al. (1996), Kumar and Byrne (2005)) allow words to change place with their local neighbors, but disallow global reorderings. Other models (Wu (1997), Xiong et al. (2006)) explicitly allow global reorderings, but do not allow all possible permutations, including some local permutations.

We present a novel technique to compare achievable translation accuracies under different reordering constraints. While earlier work has trained and tested instantiations of different reordering models and then compared the translation results (Zens and Ney, 2003) we provide a more general mechanism to evaluate the *potential* efficacy of reordering constraints, independent of specific training paradigms. Our technique attempts to answer the question: *What is the highest* BLEU *score that a given translation system could reach when using reordering constraints X?* Using this oracle approach, we abstract away from issues that are not inherent in the reordering constraints, but may nevertheless influence the comparison results, such as model and feature design, feature selection, or parameter estimation. In fact, we compare several sets of reordering constraints empirically, but do not train them as models. We merely decode by efficiently searching over possible translations allowed by each model and choosing the reordering that achieves the highest BLEU score.

We start by introducing popular reordering constraints (Section 2). Then, we present dynamic-programming algorithms that find the highest-scoring permutations of sentences under given reordering constraints (Section 3). We use this technique to compare several reordering constraints empirically. We combine a basic translation framework with different reordering constraints (Section 4) and present results on a German-English translation task (Section 5). Finally, we offer an analysis of the results and provide a review of related work (Sections 6–8).

2 Reordering Constraints

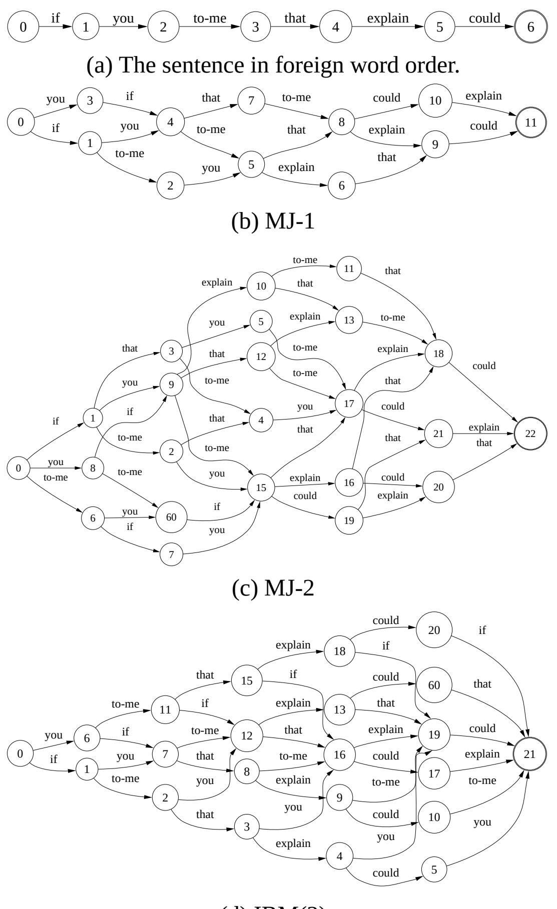
Reordering constraints restrict the movement of words or phrases in order to reach or approximate the word order of the target language. Some of the constraints considered in this paper were originally proposed for reordering words, but we will describe all constraints in terms of reordering phrases. Phrases are units of consecutive words read off a phrase translation table.

2.1 Local Constraints

Local constraints allow phrases to swap with one another only if they are adjacent or very close to each other. Kumar and Byrne (2005) define two local reordering models for their Translation Template Model (TTM): In the first one, called MJ-1, only adjacent phrases are allowed to swap, and the movement has to be done within a window of 2. A sequence consisting of three phrases *abc* can therefore become *acb* or *bac*, but not *cba*. One phrase can jump at most one phrase ahead and cannot take part in more than one swap. In their second strategy, called MJ-2, phrases are allowed to swap with their immediate neighbor or with the phrase next to the immediate neighbor; the maximum jump length is 2. This allows for all six possible permutations of *abc*. The movement here has to take place within a window of 3 phrases. Therefore, a four-phrase sequence *abcd* cannot be reordered to *cadb*, for example. MJ-1 and MJ-2 are shown in Figure 1.

2.2 IBM Constraints

First introduced by Brown et al. (1996), the IBM constraints are among the most well-known and most widely used reordering paradigms. Translation is done from the beginning of the sentence to the end, phrase by phrase; at each point in time, the constraints allow one of the first k still untranslated phrases to be selected for translation (see Figure 1d, for k=2). The IBM constraints are much less restrictive than local constraints. The first word of the input, for example, can move all the way to the end, independent of the value of k. Typically, k is set to 4 (Zens and Ney, 2003). We write IBM with k=4 as IBM(4). The IBM constraints are supersets of the local constraints.



(a) The sentence in foreign word order.

(b) MJ-1

(c) MJ-2

(d) IBM(2)

Figure 1: The German word order *if you to-me that explain could* ('wenn Sie mir das erklären könnten') and all possible reorderings under different constraints, represented as lattices. None of these lattices contains the correct English order *if you could explain that to-me*. See also Table 1.

2.3 ITG Constraints

The Inversion Transduction Grammar (ITG) (Wu, 1997), a derivative of the Syntax Directed Transduction Grammars (Aho and Ullman, 1972), constrains the possible permutations of the input string by defining rewrite rules that indicate permutations of the string. In particular, the ITG allows all permutations defined by all binary branching structures where the children of any constituent may be swapped in order. The ITG constraint is different from the other reordering constraints presented in that it is not based on finite-state operations. An

Model	# perm.	"Best" sentence	n-gram precisions	BLEU
MJ-1	13	if you that to-me could explain	100.0/66.7/20.0/0.0	0.0
MJ-2	52	to-me if you could explain that	100.0/83.3/60.0/50.0	70.71
IBM(2)	32	if to-me that you could explain	100.0/50.0/20.0/0.0	0.0
IBM(4)	384	if you could explain that to-me	100.0/100.0/100.0/100.0	100.0
IBM(4) (prune)	42	if you could explain that to-me	100.0/100.0/100.0/100.0	100.0
ITG	394	if you could explain that to-me	100.0/100.0/100.0/100.0	100.0
ITG (prune)	78	if you could explain that to-me	100.0/100.0/100.0/100.0	100.0

Table 1: Illustrating example: The number of permutations (# perm.) that different reordering paradigms consider for the input sequence *if you to-me that explain could*, and the permutation with highest BLEU score. The sentence length is 7, but there are only 6! possible permutations, since the phrase *to-me* counts as one word during reordering. ITG (prune) is the ITG BLEU decoder with the pruning settings we used in our experiments (beam threshold 10^{-4}). For comparison, IBM(4) (prune) is the lattice BLEU decoder with the same pruning settings, but we use pruning only for ITG permutations in our experiments.

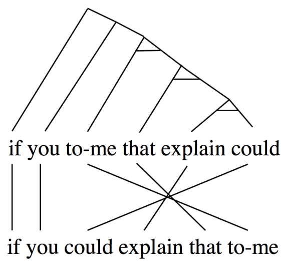

if you to-me that explain could
if you could explain that to-me

Figure 2: The example *if you to-me that explain could* and its reordering to *if you could explain that to-me* using an ITG. The alignments are added below the tree, and the horizontal bars in the tree indicate a swap.

ITG decoder runs in polynomial time and allows for long-distance phrasal reordering. A phrase can, for example, move from the first position in the input to the last position in the output and vice versa, by swapping the topmost node in the constructed binary tree. However, due to the binary bracketing constraint, some permutations are not modeled. A four-phrase sequence *abcd* cannot be permuted into *cadb* or *bdac*. Therefore, the ITG constraints are not supersets of the IBM constraints. IBM(4), for example, allows *abcd* to be permuted into *cadb* and *bdac*.

3 Factored BLEU Computation

The different reordering strategies described allow for different permutations and restrict the search space in different ways. We are concerned with the maximal achievable accuracy under given constraints, independent of feature design or parameter estimation. This is what we call the *oracle* accuracy under the reordering constraints and it is computed on a dataset with reference translations.

We now describe algorithms that can be used to find such oracle translations among unreordered translation candidates. There are two equivalent strategies: The reordering constraints that are being tested can be expressed as a special dynamic-programming decoder which, when applied to an unreordered hypothesis, searches the space of permutations defined by the reordering constraints and returns the highest-scoring permutation. We employ this strategy for the ITG reorderings (Section 3.2). For the other reordering constraints, we employ a more generic strategy: Given the set of reordering constraints, all permutations of an unreordered translation candidate are precomputed and explicitly represented as a lattice. This lattice is passed as input to a Dijkstra-style decoder (Section 3.1) which traverses it and finds the solution that reachest the highest BLEU score.[1]

3.1 Dijkstra BLEU Decoder

The Dijkstra-style decoder takes as input a lattice in which each path represents one possible permutation of an unreordered hypothesis under a given reordering paradigm, as in Figure 1. It traverses the lattice and finds the solution that has the highest approximate BLEU score, given the reference. The dynamic-programming algorithm divides the problem into subproblems that are solved independently, the solutions of which contribute to the solutions of other subproblems. The general procedure is sketched in Figure 3: for each subpath of the lattice containing the precomputed permutations, we store the three most recently attached words (Fig-

[1]For both strategies, several unreordered translation candidates do not have to be regarded separately, but can be represented as a weighted lattice and be used as input to the special dynamic program or to the process that precomputes possible permutations.

$$\beta([0,k,len+1,w_2,w_3,w_{new}]) = \max_{w_1} (\text{ get\_bleu } ([0,j,len,w_1,w_2,w_3],[j,k,w_{new}])) \tag{1}$$

```
function get_bleu ( $[0,j,len,w_1,w_2,w_3],[j,k,w_{new}]$ ) :=
    update_ngrams $(0,j,k,len,w_1,w_2,w_3,w_{new})$ ;
    return $exp\left(\frac{1}{4}\sum_{n=1}^{4} log\left(\frac{\text{ngrams}_i([0,k,len+1,w_2,w_3,w_{new}])}{len-n+1}\right)\right)$ ;          (2)
```

Figure 3: **Top:** The BLEU score is used as inside score for a subpath from 0 to k with the rightmost words w_2, w_3, w_{new} in the Dijkstra decoder. **Bottom:** Pseudo code for a function `get_bleu` which updates the n-gram matches ngrams$_1(\ldots)$, ngrams$_2(\ldots)$, ngrams$_3(\ldots)$, ngrams$_4(\ldots)$ for the resulting subpath in a hash table $[0,k,len+1,w_2,w_3,w_{new}]$ and returns its approximate BLEU score.

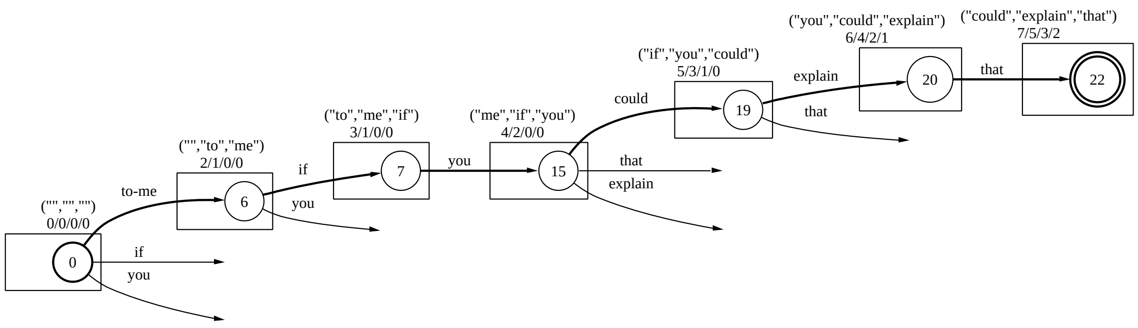


Figure 4: Three right-most words and n-gram matches: This shows the best path for the MJ-2 reordering of *if you to-me that explain could*, along with the words stored at each state and the progressively updated n-gram matches. The full path *to-me if you could explain that* has 7 unigram matches, 5 bigram, 3 trigram, and 2 fourgram matches. See the full MJ-2 lattice in Figure 1c.

ure 4). A context of three words is needed to compute fourgram precisions used in the BLEU score. Starting from the start state, we recursively extend a subpath word by word, following the paths in the lattice. Whenever we extend the path by a word to the right we incorporate that word and use `update_ngrams` to update the four n-gram counts for the subpath. The function `update_ngrams` has access to the reference string[2] and stores the updated n-gram counts for the resulting path in a hash table.[3] The inside score of each subpath is the approximate BLEU score, calculated as the average of the four n-gram log precisions. An n-gram precision is always the number of n-gram matches divided by the length len of the path minus $(n-1)$. A path of length 4 with 2 bigram matches, for example, has a bigram precision of $^2/_3$. This method is similar to Dijkstra's algorithm (Dijkstra, 1959) composed with a fourgram finite-state language model, where the scoring is done using n-gram counts and precision scores. We call this the Dijkstra BLEU decoder.

3.2 ITG BLEU Decoder

For the ITG reordering constraints, we use a dynamic program that computes the permutations implicitly. It takes only the unreordered hypothesis as input and creates the possible reorderings under the ITG constraints during decoding, as it creates a parse chart. The algorithm is similar to a CKY parsing algorithm in that it proceeds bottom-up and combines smaller constituents into larger ones recursively. Figure 5 contains details of the algorithm. The ITG BLEU decoder stores the three leftmost and the three rightmost words in each constituent. A constituent from position i to position k, with w_a, w_b, and w_c as leftmost words, and w_x, w_y, w_z as rightmost words is written as $[i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]$. Such a constituent can be built by straight or inverted rules. Using an inverted rule means swapping the order of the children in the built constituent. The successive bottom-up combinations of adjacent constituents result in hierarchical binary bracketing with swapped and non-

[2]Multiple reference strings can be used if available.

[3]An epsilon value of 1^{-10} is used for zero precisions.

$$\beta\left([i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]\right) = \max \begin{pmatrix} \beta_{()}\left([i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]\right), \\ \beta_{<>}\left([i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]\right) \end{pmatrix} \tag{3}$$

$$\beta_{<>}([i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]) = \max_{j,w_{a'},w_{b'},w_{c'},w_{x'},w_{y'},w_{z'}} \left(\text{get\_bleu} \begin{pmatrix} [j,k,(w_a,w_b,w_c),(w_{x'},w_{y'},w_{z'})], \\ [i,j,(w_{a'},w_{b'},w_{c'}),(w_x,w_y,w_z)] \end{pmatrix} \right) \tag{4}$$

Figure 5: Equations for the ITG oracle BLEU decoder. $[i,k,(w_a,w_b,w_c),(w_x,w_y,w_z)]$ is a constituent from i to k with leftmost words w_a,w_b,w_c and rightmost words w_x,w_y,w_z. Top: A constituent can be built with a straight or a swapped rule. Bottom: A swapped rule. The `get_bleu` function can be adapted from Figure 3

swapped constituents. Our ITG BLEU decoder uses standard beam search pruning. As in Zens and Ney (2003), phrases are not broken up, but every phrase is, at the beginning of reordering, stored in the chart as one lexical token together with the precomputed n-gram matches and the n-gram precision score.

In addition to standard ITG we run experiments with a constrained ITG, in which we impose a bound ρ on the maximum length of reordered constituents, measured in phrases. If the combined length of two constituents exceeds this bound they can only be combined in the given monotone order. Experiments with this ITG variant give insight into the effect that various long-distance reorderings have on the final BLEU scores (see Table 3). Such bounds are also effective speedup techniques(Eisner and Tromble, 2006).

3.3 BLEU Approximations

BLEU is defined to use the *modified* n-gram precision, which means that a correct n-gram that occurs once in the reference, but several times in the system translation will be counted only once as correct. The other occurrences are clipped. We do not include this global feature since we want a dynamic-programming solution with polynomial size and runtime. The decoder processes subproblems independently; words are attached locally and stored only as boundary words of covered paths/ constituents. Therefore we cannot discount a locally attached word that has already been attached elsewhere to an alternative path/constituent. However, clipping affects most heavily the unigram scores which are constant, like the length of the sentence.[4]

We also adopt the approximation that treats every sentence with its reference as a separate corpus (Tillmann and Zhang, 2006) so that ngram counts are not accumulated, and parallel processing of sentences becomes possible. Due to these two approximations, our method is not guaranteed to find the best reordering defined by the reordering constraints. However, we have found on our heldout data that an oracle that does not accumulate n-gram counts is only minimally worse than an oracle that does accumulate them (up to 0.25 BLEU points).[5] If, in addition, clipping is ignored, the resulting oracle stays virtually the same, at most 0.02 BLEU points worse than the oracle found otherwise. All results in this paper are computed with the original BLEU formula on the sentences found by the oracle algorithms.

4 Creating a Monotone Translation Baseline

To compare the reordering constraints under oracle conditions we first obtain unreordered candidate translations from a simple baseline translation model. For each reordering paradigm, we take the candidate translations, get the best oracle reorderings under the given reordering constraints and pick the best sentence according to the BLEU score.

The baseline translation system is created using probabilistic word-to-word and phrase-to-phrase ta-

[4]Since the sentence lengths are constant for all reorderings of a given sentence we can in our experiments also ignore the brevity penalty which cancels out. If the input consists of several sentences of different lengths (see fn. 1) then the brevity penalty can be built in by keeping track of length ratios of attached phrases.

[5]The accumulating oracle algorithm makes a greedy decision for every sentence given the ngram counts so far accumulated (Zens and Ney, 2005). The result of such a greedy oracle method may depend on the order of the input sentences. We tried 100 shuffles of these and received 100 very similar results, with a variance of under 0.006 BLEU points. The non-accumulating oracles use an epsilon value (1^{-10}) for zero counts.

bles. Using the translation probabilities, we create a lattice that contains word and phrase translations for every substring of the source sentence. The resulting lattice is made of English words and phrases of different lengths. Every word or phrase translation probability p is a mixture of $p(f|e)$ and $p(e|f)$. We discard short phrase translations exponentially by a parameter that is trained on heldout data. Insertions and deletions are handled exclusively by the use of a phrase table: an insertion takes place wherever the English side of a phrase translation is longer than the foreign side (e.g. English *presidential candidate* for German *Präsidentschaftskandidat*), and vice versa for deletions (e.g. *we discussed* for *wir haben diskutiert*). Gaps or discontinuous phrases are not handled. The baseline decoder outputs the n-best paths through the lattice according to the lattice scores[6], marking consecutive phrases so that the oracle reordering algorithms can recognize them and keep them together. Note that the baseline system is trained on real data, while the reordering constraints that we want to test are not trained.

5 Empirical Comparison of Reordering Constraints

We use the monotone translation baseline model and the oracle BLEU computation to evaluate different popular reordering strategies. We now describe the experimental settings. The word and phrase translation probabilities of the baseline model are trained on the Europarl German-English training set, using GIZA++ and the Pharaoh phrase extraction algorithm. For testing we use the NAACL 2006 SMT Shared Task test data. For each sentence of the test set, a lattice is created in the way described in Section 4, with parameters optimized on a small heldout set.[7] For each sentence, the 1000-best candidates according to the lattice scores are extracted. We take the 10-best oracle candidates, according to the reference, and use a BLEU decoder to create the best permutation of each of them and pick the best one. Using this procedure, we make sure that we get the highest-scoring unreordered candidates and choose the best one among their oracle reorderings. Table 2 and Figure 6 show the resulting BLEU scores for different sentence lengths. Table 3 shows results of the ITG runs with different length bounds ρ. The average phrase length in the candidate translations of the test set is 1.42 words.

Oracle decodings under the ITG and under IBM(4) constraints were up to 1000 times slower than under the other tested oracle reordering methods in our implementations. Among the faster methods, decoding under MJ-2 constraints was up to 40% faster than under IBM(2) constraints in our implementation.

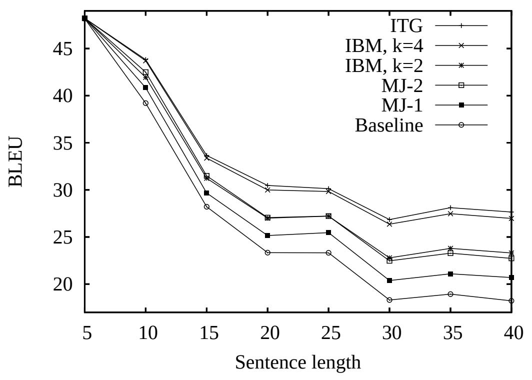


Figure 6: Reordering oracle scores for different sentence lengths. See also Table 2.

6 Discussion

The empirical results show that reordering under sufficiently permissive constraints can improve a monotone baseline oracle by more than 7.5 BLEU points. This gap between choosing the best unreordered sentences versus choosing the best optimally reordered sentences is small for short sentences and widens dramatically (more than nine BLEU points) for longer sentences.

The ITG constraints and the IBM(4) constraints both give very high oracle translation accuracies on the German-English translation task. Overall, their BLEU scores are about 2 to more than 4 points better than the BLEU scores of the best other methods. This gap between the two highest-scoring constraints and the other methods becomes bigger as the sentence lengths grow and is greater than 4

[6]We use a straightforward adaption of Algorithm 3 in Huang and Chiang (2005)

[7]We fill the initial phrase and word lattice with the 20 best candidates, using phrases of 3 or less words.

Sentence length	# of test sentences	ITG (prune)	IBM, k=4	IBM, k=2	MJ-2	MJ-1	No reordering
		BLEU (NIST) scores					
1–5	61	48.21 (5.35)	48.21 (5.35)	48.21 (5.35)	48.21 (5.35)	48.21 (5.35)	48.17 (5.68)
6–10	230	43.83 (6.75)	43.71 (6.74)	41.94 (6.68)	42.50 (6.71)	40.85 (6.66)	39.21 (6.99)
11–15	440	33.66 (6.71)	33.37 (6.71)	31.23 (6.62)	31.49 (6.64)	29.67 (6.56)	28.21 (6.76)
16–20	447	30.47 (6.66)	29.99 (6.65)	27.00 (6.52)	27.06 (6.50)	25.15 (6.45)	23.34 (6.52)
21–25	454	30.13 (6.80)	29.83 (6.79)	27.21 (6.67)	27.22 (6.65)	25.46 (6.58)	23.32 (6.63)
26–30	399	26.85 (6.42)	26.36 (6.42)	22.79 (6.25)	22.47 (6.22)	20.38 (6.12)	18.31 (6.11)
31–35	298	28.11 (6.45)	27.47 (6.43)	23.79 (6.25)	23.28 (6.21)	21.09 (6.12)	18.94 (6.06)
36–40	242	27.65 (6.37)	26.97 (6.35)	23.31 (6.19)	22.73 (6.16)	20.70 (6.06)	18.22 (5.94)
1–40	2571	29.63 (7.48)	29.17 (7.46)	26.07 (7.24)	25.89 (7.22)	23.95 (7.08)	21.89 (7.07)

Table 2: BLEU and NIST results for different reordering methods on binned sentence lengths. The ITG results are, unlike the other results, with pruning (beam 10^{-4}). The BLEU results are plotted in Figure 6. All results are computed with the original BLEU formula on the sentences found by the oracle algorithms.

BLEU scores for sentences longer than 30 sentences. This advantage in translation accuracy comes with high computational cost, as mentioned above.

Among the computationally more lightweight reordering methods tested, IBM(2) and MJ-2 are very close to each other in translation accuracy, with IBM(2) obtaining slightly better scores on longer sentences, while MJ-2 is more efficient. MJ-1 is less successful in reordering, improving the monotone baseline by only about 2.5 BLEU points at best, but is the best choice if speed is an issue.

As described above, the reorderings defined by the local constraints MJ-1 and MJ-2 are subsets of IBM(2) and IBM(3). We did not test IBM(3), but the values can be interpolated between IBM(2) and IBM(4). The ITG constraints do not belong in this family of finite-state contraints; they allow reorderings that none of the other methods allow, and vice versa. The fact that ITG constraints can reach such high translation accuracies supports the findings in Zens et al. (2004) and is an empirical validation of the ITG hypothesis.

The experiments with the constrained ITG show the effect of reorderings spanning different lengths (see Table 3). While most reorderings are short-distance ($<$5 phrases) a lot of improvements can still be obtained when ρ is increased from length 5 to 10 and even from 10 to 20 phrases.

Len.	ρ=0	ρ=5	ρ=10	ρ=20	ρ=30	ρ=40
26–30	18.31	24.07	26.40	26.79	26.85	26.85
31–35	18.94	25.10	27.21	28.00	28.09	28.11
36–40	18.22	24.46	26.66	27.53	27.64	27.65
26–40	18.49	24.74	26.74	27.41	27.50	27.51

Table 3: BLEU results of ITGs that are constrained to reorderings not exceeding a certain span length ρ. Results shown for different sentence lengths.

7 Related Work

There exist related algorithms that search the space of reorderings and compute BLEU oracle approximations. Zens and Ney (2005) describe a dynamic-programming algorithm in which at every state the number of n-gram matches is stored, along with a multiset that contains all words from the reference that have not yet been matched. This makes it possible to compute the *modified* ngram precision, but the search space is exponential. Tillmann and Zhang (2006) use a BLEU oracle decoder for discriminative training of a local reordering model. No details about the algorithm are given. Zens and Ney (2003) perform a comparison of different reordering strategies. Their study differs from ours in that they use reordering models trained on real data and may therefore be influenced by feature selection, parameter estimation and other training-specific issues. In our study, only the baseline translation model is trained on data. Zens et al. (2004) conduct a study similar to Zens and Ney (2003) and note that the results for the ITG reordering constraints were quite dependent on the very simple probability model used. Our study avoids this issue by using the

BLEU oracle approach. In Wellington et al. (2006), hand-aligned data are used to compare the standard ITG constraints to ITGs that allow gaps.

8 Conclusions

We have presented a training-independent method to compare different reordering constraints for machine translation. Given a sentence in foreign word order, its reference translation(s) and reordering constraints, our dynamic-programming algorithms efficiently find the oracle reordering that has the approximately highest BLEU score. This allows evaluating different reordering constraints experimentally, but abstracting away from specific features, the probability model or training methods of the reordering strategies. The presented method evaluates the theoretical capabilities of reordering constraints, as opposed to more arbitrary accuracies of specifically trained instances of reordering models.

Using our oracle method, we presented an empirical evaluation of different reordering constraints for a German-English translation task. The results show that a good reordering of a given monotone translation can improve the translation quality dramatically. Both short- and long-distance reorderings contribute to the BLEU score improvements, which are generally greater for longer sentences. Reordering constraints that allow global reorderings tend to reach better oracles scores than ones that search more locally. The ITG constraints and the IBM(4) constraints both give the highest oracle scores.

The presented BLEU decoder algorithms can be useful in many ways: They can generally help decide what reordering constraints to choose for a given translation system. They can be used for discriminative training of reordering models (Tillmann and Zhang, 2006). Furthermore, they can help detecting insufficient parameterization or incapable training algorithms: If two trained reordering model instances show similar performances on a given task, but the oracle scores differ greatly then the training methods might not be optimal.

Acknowledgments

This work was partially supported by the National Science Foundation via an ITR grant (No 0121285), the Defense Advanced Research Projects Agency via a GALE contract (No HR0011-06-2-0001), and the Office of Naval Research via a MURI grant (No N00014-01-1-0685). We thank Jason Eisner, David Smith, Roy Tromble and the anonymous reviewers for helpful comments and suggestions.

References

A. V. Aho and J. D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*. Prentice Hall.

A.L. Berger P. F. Brown, S. A. Della Pietra, V. J. Della Pietra, J. R. Gillett, J. D. Lafferty, R. L. Mercer, H. Printz, and L. Ures. 1996. Language translation apparatus and method using context-based translation models. United States Patent No. 5,510,981.

E.W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik.*, 1:269–271.

J. Eisner and R. W. Tromble. 2006. Local search with very large-scale neighborhoods for optimal permutations in Machine Translation. In *Proc. of the Workshop on Computationally Hard Problems and Joint Inference*, New York.

L. Huang and D. Chiang. 2005. Better k-best parsing. In *Proc. of IWPT*, Vancouver, B.C., Canada.

S. Kumar and W. Byrne. 2005. Local phrase reordering models for Statistical Machine Translation. In *Proc. of HLT/EMNLP*, pages 161–168, Vancouver, B.C., Canada.

C. Tillmann and T. Zhang. 2006. A discriminative global training algorithm for Statistical MT. In *Proc. of ACL*, pages 721–728, Sydney, Australia.

B. Wellington, S. Waxmonsky, and D. Melamed. 2006. Empirical lower bounds on the complexity of translational equivalence. In *Proc. of COLING-ACL*, pages 977–984, Sydney, Australia.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):377–404.

D. Xiong, Q. Liu, and S. Lin. 2006. Maximum entropy based phrase reordering model for Statistical Machine Translation. In *Proc. of COLING-ACL*, pages 521–528, Sydney, Australia.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in Statistical Machine Translation. In *Proc. of ACL*, pages 144–151, Sapporo, Japan.

R. Zens and H. Ney. 2005. Word graphs for Statistical Machine Translation. In *Proc. of the ACL Workshop on Building and Using Parallel Texts*, pages 191–198, Ann Arbor, MI.

R. Zens, H. Ney, T. Watanabe, and E. Sumita. 2004. Reordering constraints for phrase-based Statistical Machine Translation. In *Proc. of CoLing*, pages 205–211, Geneva.