A Comparison of Syntactically Motivated Word Alignment Spaces

Colin Cherry
Department of Computing Science
University of Alberta
Edmonton, AB, Canada, T6G 2E8
colinc@cs.ualberta.ca

Dekang Lin
Google Inc.
1600 Amphitheatre Parkway
Mountain View, CA, USA, 94043
lindek@google.com

Abstract

This work is concerned with the space of alignments searched by word alignment systems. We focus on situations where word re-ordering is limited by syntax. We present two new alignment spaces that limit an ITG according to a given dependency parse. We provide D-ITG grammars to search these spaces completely and without redundancy. We conduct a careful comparison of five alignment spaces, and show that limiting search with an ITG reduces error rate by 10%, while a D-ITG produces a 31% reduction.

1 Introduction

Bilingual word alignment finds word-level correspondences between parallel sentences. The task originally emerged as an intermediate result of training the IBM translation models (Brown et al., 1993). These models use minimal linguistic intuitions; they essentially treat sentences as flat strings. They remain the dominant method for word alignment (Och and Ney, 2003). There have been several proposals to introduce syntax into word alignment. Some work within the framework of synchronous grammars (Wu, 1997; Melamed, 2003), while others create a generative story that includes a parse tree provided for one of the sentences (Yamada and Knight, 2001).

There are three primary reasons to add syntax to word alignment. First, one can incorporate syntactic features, such as grammar productions, into the models that guide the alignment search. Second, movement can be modeled more naturally; when a three-word noun phrase moves during translation, it can be modeled as one movement operation instead of three. Finally, one can restrict the type of movement that is considered, shrinking the number of alignments that are attempted. We investigate this last advantage of syntactic alignment. We fix an alignment scoring model that works equally well on flat strings as on parse trees, but we vary the space of alignments evaluated with that model. These spaces become smaller as more linguistic guidance is added. We measure the benefits and detriments of these constrained searches.

Several of the spaces we investigate draw guidance from a dependency tree for one of the sentences. We will refer to the parsed language as English and the other as Foreign. Lin and Cherry (2003) have shown that adding a dependency-based cohesion constraint to an alignment search can improve alignment quality. Unfortunately, the usefulness of their beam search solution is limited: potential alignments are constructed explicitly, which prevents a perfect search of alignment space and the use of algorithms like EM. However, the cohesion constraint is based on a tree, which should make it amenable to dynamic programming solutions. To enable such techniques, we bring the cohesion constraint inside the ITG framework (Wu, 1997).

Zhang and Gildea (2004) compared Yamada and Knight's (2001) tree-to-string alignment model to ITGs. They concluded that methods like ITGs, which create a tree during alignment, perform better than methods with a fixed tree established before alignment begins. However, the use of a fixed tree is not the only difference between (Yamada and Knight, 2001) and ITGs; the probability models are also very different. By using a fixed dependency tree inside an ITG, we can revisit the question of whether using a fixed tree is harmful, but in a controlled environment.

2 Alignment Spaces

Let an **alignment** be the entire structure that connects a sentence pair, and let a **link** be the individual word-to-word connections that make up an alignment. An **alignment space** determines the set of all possible alignments that can exist for a given sentence pair. Alignment spaces can emerge from generative stories (Brown et al., 1993), from syntactic notions (Wu, 1997), or they can be imposed to create competition between links (Melamed, 2000). They can generally be described in terms of how links interact.

For the sake of describing the size of alignment spaces, we will assume that both sentences have n tokens. The largest alignment space for a sentence pair has 2^{n^2} possible alignments. This describes the case where each of the n^2 potential links can be either on or off with no restrictions.

2.1 Permutation Space

A straight-forward way to limit the space of possible alignments is to enforce a one-to-one constraint (Melamed, 2000). Under such a constraint, each token in the sentence pair can participate in at most one link. Each token in the English sentence picks a token from the Foreign sentence to link to, which is then removed from competition. This allows for $n!$ possible alignments[1], a substantial reduction from 2^{n^2}.

Note that $n!$ is also the number of possible permutations of the n tokens in either one of the two sentences. **Permutation space** enforces the one-to-one constraint, but allows any reordering of tokens as they are translated. Permutation space methods include weighted maximum matching (Taskar et al., 2005), and approximations to maximum matching like competitive linking (Melamed, 2000). The IBM models (Brown et al., 1993) search a version of permutation space with a one-to-many constraint.

2.2 ITG Space

Inversion Transduction Grammars, or ITGs (Wu, 1997) provide an efficient formalism to synchronously parse bitext. This produces a parse tree that decomposes both sentences and also implies a word alignment. ITGs are transduction grammars because their terminal symbols can produce tokens in both the English and Foreign sentences. Inversions occur when the order of constituents is reversed in one of the two sentences.

In this paper, we consider the alignment space induced by parsing with a binary bracketing ITG, such as:

$$A \rightarrow [AA] \mid \langle AA \rangle \mid e/f \qquad (1)$$

The terminal symbol e/f represents tokens output to the English and Foreign sentences respectively. Square brackets indicate a straight combination of non-terminals, while angle brackets indicate an inverted combination: $\langle A_1 A_2 \rangle$ means that $A_1 A_2$ appears in the English sentence, while $A_2 A_1$ appears in the Foreign sentence.

Used as a word aligner, an ITG parser searches a subspace of permutation space: the ITG requires that any movement that occurs during translation be explained by a binary tree with inversions. Alignments that allow no phrases to be formed in bitext are not attempted. This results in two forbidden alignment structures, shown in Figure 1, called "inside-out" transpositions in (Wu, 1997). Note that no pair of contiguous tokens in the top sentence remain contiguous when projected onto the bottom sentence. Zens and Ney (2003) explore the re-orderings allowed by ITGs, and provide a formulation for the number of structures that can be built for a sentence pair of size n. ITGs explore almost all of permutation space when n is small, but their coverage of permutation space falls off quickly for $n > 5$ (Wu, 1997).

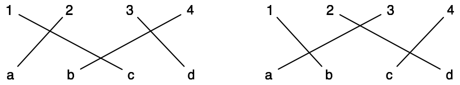

Figure 1: Forbidden alignments in ITG

2.3 Dependency Space

Dependency space defines the set of all alignments that maintain phrasal cohesion with respect to a dependency tree provided for the English sentence. The space is constrained so that the phrases in the dependency tree always move together.

Fox (2002) introduced the notion of head-modifier and modifier-modifier crossings. These occur when a phrase's image in the Foreign sentence overlaps with the image of its head, or one of its siblings. An alignment with no crossings maintains phrasal cohesion. Figure 2 shows a head-modifier crossing: the image c of a head 2 overlaps with the image (b, d) of 2's modifier, $(3, 4)$. Lin and Cherry (2003) used the notion of phrasal cohe-

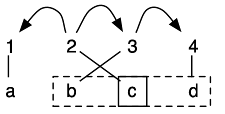

Figure 2: A phrasal cohesion violation.

[1] This is a simplification that ignores *null* links. The actual number of possible alignments lies between $n!$ and $(n+1)^n$.

sion to constrain a beam search aligner, conducting a heuristic search of the dependency space.

The number of alignments in dependency space depends largely on the provided dependency tree. Because all permutations of a head and its modifiers are possible, a tree that has a single head with $n-1$ modifiers provides no guidance; the alignment space is the same as permutation space. If the tree is a chain (where every head has exactly one modifier), alignment space has only 2^n permutations, which is by far the smallest space we have seen. In general, there are $\prod_\theta [(m_\theta + 1)!]$ permutations for a given tree, where θ stands for a head node in the tree, and m_θ counts θ's modifiers. Dependency space is not a subspace of ITG space, as it can create both the forbidden alignments in Figure 1 when given a single-headed tree.

3 Dependency constrained ITG

In this section, we introduce a new alignment space defined by a dependency constrained ITG, or D-ITG. The set of possible alignments in this space is the intersection of the dependency space for a given dependency tree and ITG space. Our goal is an alignment search that respects the phrases specified by the dependency tree, but attempts all ITG orderings of those phrases, rather than all permutations. The intuition is that most ordering decisions involve only a small number of phrases, so the search should still cover a large portion of dependency space.

This new space has several attractive computational properties. Since it is a subspace of ITG space, we will be able to search the space completely using a polynomial time ITG parser. This places an upper bound on the search complexity equal to ITG complexity. This upper bound is very loose, as the ITG will often be drastically constrained by the phrasal structure of the dependency tree. Also, by working in the ITG framework, we will be able to take advantage of advances in ITG parsing, and we will have access to the forward-backward algorithm to implicitly count events over all alignments.

3.1 A simple solution

Wu (1997) suggests that in order to have an ITG take advantage of a known partial structure, one can simply stop the parser from using any spans that would violate the structure. In a chart parsing framework, this can be accomplished by assigning the invalid spans a value of $-\infty$ before parsing begins. Our English dependency tree qualifies as a partial structure, as it does not specify a complete binary decomposition of the English sentence. In this case, any ITG span that would contain part, but not all, of two adjacent dependency phrases can be invalidated. The sentence pair can then be parsed normally, automatically respecting phrases specified by the dependency tree.

For example, Figure 3a shows an alignment for the sentence pair, “His house in Canada, Sa maison au Canada” and the dependency tree provided for the English sentence. The spans disallowed by the tree are shown using underlines. Note that the illegal spans are those that would break up the “in Canada” subtree. After invalidating these spans in the chart, parsing the sentence pair with the bracketing ITG in (1) will produce the two structures shown in Figure 3b, both of which correspond to the correct alignment.

This solution is sufficient to create a D-ITG that obeys the phrase structure specified by a dependency tree. This allows us to conduct a complete search of a well-defined subspace of the dependency space described in Section 2.3.

3.2 Avoiding redundant derivations with a recursive ITG

The above solution can derive two structures for the same alignment. It is often desirable to eliminate redundant structures when working with ITGs. Having a single, canonical tree structure for each possible alignment can help when flattening binary trees, as it indicates arbitrary binarization decisions (Wu, 1997). Canonical structures also eliminate double counting when performing tasks like EM (Zhang and Gildea, 2004). The nature of *null* link handling in ITGs makes eliminating all redundancies difficult, but we can at least eliminate them in the absence of *nulls*.

Normally, one would eliminate the redundant structures produced by the grammar in (1) by replacing it with the canonical form grammar (Wu, 1997), which has the following form:

$$\begin{array}{rl} S \rightarrow & A \mid B \mid C \\ A \rightarrow & [AB] \mid [BB] \mid [CB] \mid \\ & [AC] \mid [BC] \mid [CC] \\ B \rightarrow & \langle AA \rangle \mid \langle BA \rangle \mid \langle CA \rangle \mid \\ & \langle AC \rangle \mid \langle BC \rangle \mid \langle CC \rangle \\ C \rightarrow & e/f \end{array} \tag{2}$$

By design, this grammar allows only one struc-

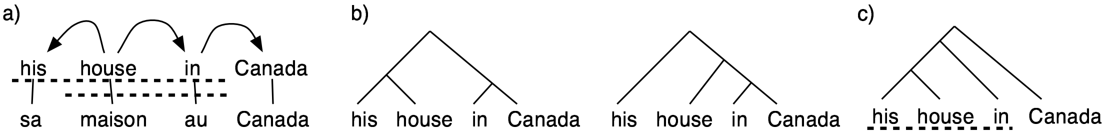

Figure 3: An example of how dependency trees interact with ITGs. (a) shows the input, dependency tree, and alignment. Invalidated spans are underlined. (b) shows valid binary structures. (c) shows the canonical ITG structure for this alignment.

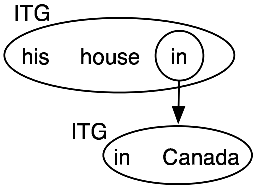

Figure 4: A recursive ITG.

ture per alignment. It works by restricting right-recursion to specific inversion combinations.

The canonical structure for a given alignment is fixed by this grammar, without awareness of the dependency tree. When the dependency tree invalidates spans that are used in canonical structures, the parser will miss the corresponding alignments. The canonical structure corresponding to the correct alignment in our running example is shown in Figure 3c. This structure requires the underlined invalid span, so the canonical grammar fails to produce the correct alignment. Our task requires a new canonical grammar that is aware of the dependency tree, and will choose among valid structures deterministically.

Our ultimate goal is to fall back to ITG reordering when the dependency tree provides no guidance. We can implement this notion directly with a recursive ITG. Let a **local tree** be the tree formed by a head node and its immediate modifiers. We begin our recursive process by considering the local tree at the root of our dependency tree, and marking each phrasal modifier with a labeled placeholder. We then create a string by flattening the local tree. The top oval of Figure 4 shows the result of this operation on our running example. Because all phrases have been collapsed to placeholders, an ITG built over this string will naturally respect the dependency tree's phrasal boundaries. Since we do not need to invalidate any spans, we can parse this string using the canonical ITG in (2). The phrasal modifiers can in turn be processed by applying the same algorithm recursively to their root nodes, as shown in the lower oval of Figure 4. This algorithm will explore the exact same alignment space as the solution presented in Section 3.1, but because it uses a canonical ITG at every ordering decision point, it will produce exactly one structure for each alignment. Returning to our running example, the algorithm will produce the left structure of Figure 3b.

This recursive approach can be implemented inside a traditional ITG framework using grammar templates. The templates take the form of whatever grammar will be used to permute the local trees. They are instantiated over each local tree before ITG parsing begins. Each instantiation has its non-terminals marked with its corresponding span, and its pre-terminal productions are customized to match the modifiers of the local tree. Phrasal modifiers point to another instantiation of the template. In our case, the template corresponds to the canonical form grammar in (2). The result of applying the templates to our running example is:

$$\begin{array}{rl}
S_{0,4} \rightarrow & A_{0,4} \mid B_{0,4} \mid C_{0,4} \\
A_{0,4} \rightarrow & [A_{0,4}B_{0,4}] \mid [B_{0,4}B_{0,4}] \mid [C_{0,4}B_{0,4}] \mid \\
 & [A_{0,4}C_{0,4}] \mid [B_{0,4}C_{0,4}] \mid [C_{0,4}C_{0,4}] \\
B_{0,4} \rightarrow & \langle A_{0,4}A_{0,4}\rangle \mid \langle B_{0,4}A_{0,4}\rangle \mid \langle C_{0,4}A_{0,4}\rangle \mid \\
 & \langle A_{0,4}C_{0,4}\rangle \mid \langle B_{0,4}C_{0,4}\rangle \mid \langle C_{0,4}C_{0,4}\rangle \\
C_{0,4} \rightarrow & \text{his}/f \mid \text{house}/f \mid S_{2,4}
\end{array}$$

$$\begin{array}{rl}
S_{2,4} \rightarrow & A_{2,4} \mid B_{2,4} \mid C_{2,4} \\
A_{2,4} \rightarrow & [A_{2,4}B_{2,4}] \mid [B_{2,4}B_{2,4}] \mid [C_{2,4}B_{2,4}] \mid \\
 & [A_{2,4}C_{2,4}] \mid [B_{2,4}C_{2,4}] \mid [C_{2,4}C_{2,4}] \\
B_{2,4} \rightarrow & \langle A_{2,4}A_{2,4}\rangle \mid \langle B_{2,4}A_{2,4}\rangle \mid \langle C_{2,4}A_{2,4}\rangle \mid \\
 & \langle A_{2,4}C_{2,4}\rangle \mid \langle B_{2,4}C_{2,4}\rangle \mid \langle C_{2,4}C_{2,4}\rangle \\
C_{2,4} \rightarrow & \text{in}/f \mid \text{Canada}/f
\end{array}$$

Recursive ITGs and grammar templates provide a conceptual framework to easily transfer grammars for flat sentence pairs to situations with fixed phrasal structure. We have used the framework here to ensure only one structure is constructed for each possible alignment. We feel that this recursive view of the solution also makes it easier to visualize the space that the D-ITG is searching. It is trying all ITG orderings of each head and its modifiers.

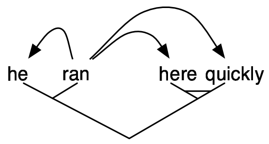

Figure 5: A counter-intuitive ITG structure.

3.3 Head constrained ITG

D-ITGs can construct ITG structures that do not completely agree with the provided dependency tree. If a head in the dependency tree has more than one modifier on one of its sides, then those modifiers may form a phrase in the ITG that should not exist according to the dependency tree. For example, the ITG structure shown in Figure 5 will be considered by our D-ITG as it searches alignment space. The resulting "here quickly" subtree disagrees with our provided dependency tree, which specifies that "ran" is modified by each word individually, and not by a phrasal concept that includes both. This is allowed by the parser because we have made the ITG aware of the dependency tree's phrasal structure, but it still has no notion of heads or modifiers. It is possible that by constraining our ITG according to this additional syntactic information, we can provide further guidance to our alignment search.

The simplest way to eliminate these modifier combinations is to parse with the redundant bracketing grammar in (1), and to add another set of invalid spans to the set described in Section 3.1. These new invalidated chart entries eliminate all spans that include two or more modifiers without their head. With this solution, the structure in Figure 5 is no longer possible. Unfortunately, the grammar allows multiple structures for each alignment: to represent an alignment with no inversions, this grammar will produce all three structures shown in Figure 6.

If we can develop a grammar that will produce canonical head-aware structures for local trees, we can easily extend it to complete dependency trees using the concept of recursive ITGs. Such a grammar requires a notion of head, so we can ensure that every binary production involves the head or a phrase containing the head. A redundant, head-aware grammar is shown here:

$$\begin{aligned} &A \rightarrow [MA] \mid \langle MA\rangle \mid [AM] \mid \langle AM\rangle \mid H \\ &M \rightarrow \text{he}/f \mid \text{here}/f \mid \text{quickly}/f \\ &H \rightarrow \text{ran}/f \end{aligned} \tag{3}$$

Note that two modifiers can never be combined without also including the A symbol, which always contains the head. This grammar still considers all the structures shown in Figure 6, but it requires no chart preprocessing.

We can create a redundancy-free grammar by expanding (3). Inspired by Wu's canonical form grammar, we will restrict the productions so that certain structures are formed only when needed for specific inversion combinations. To specify the necessary inversion combinations, our ITG will need more expressive non-terminals. Split A into two non-terminals, L and R, to represent generators for left modifiers and right modifiers respectively. Then split L into $\bar{L}$ and $\hat{L}$, for generators that produce straight and inverted left modifiers.

We now have a rich enough non-terminal set to design a grammar with a default behavior: it will generate all right modifiers deeper in the bracketing structure than all left modifiers. This rule is broken only to create a re-ordering that is not possible with the default structure, such as $[\langle MH\rangle\, M]$. A grammar that accomplishes this goal is shown here:

$$\begin{aligned} &S \rightarrow \bar{L}|\hat{L}|R \\ &R \rightarrow \left[\hat{L}M\right] \mid \langle\bar{L}M\rangle \mid [RM] \mid \langle RM\rangle \mid H \\ &\bar{L} \rightarrow [M\bar{L}] \mid \left[M\hat{L}\right] \mid [MR] \\ &\hat{L} \rightarrow \langle M\bar{L}\rangle \mid \left\langle M\hat{L}\right\rangle \mid \langle MR\rangle \\ &M \rightarrow \text{he}/f \mid \text{here}/f \mid \text{quickly}/f \\ &H \rightarrow \text{ran}/f \end{aligned} \tag{4}$$

This grammar will generate one structure for each alignment. In the case of an alignment with no inversions, it will produce the tree shown in Figure 6c. The grammar can be expanded into a recursive ITG by following a process similar to the one explained in Section 3.2, using (4) as a template.

3.3.1 The head-constrained alignment space

Because we have limited the ITG's ability to combine them, modifiers of the same head can no longer occur at the same level of any ITG tree. In Figure 6, we see that in all three valid structures, "quickly" is attached higher in the tree than "here". As a result of this, no combination of inversions can bring "quickly" between "here" and "ran". In general, the alignment space searched by this ITG is constrained so that, among modifiers, relative distance from head is maintained. More formally, let M_i and M_o be modifiers of H such that M_i appears between M_o and H in the dependency tree. No alignment will ever place the

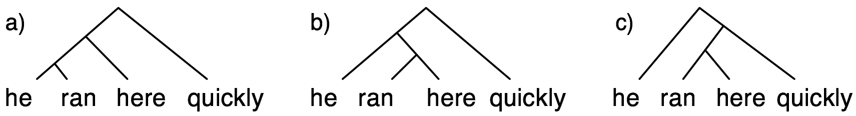

Figure 6: Structures allowed by the head constraint.

outer modifier M_o between H and the inner modifier M_i.

4 Experiments and Results

We compare the alignment spaces described in this paper under two criteria. First we test the guidance provided by a space, or its capacity to stop an aligner from selecting bad alignments. We also test expressiveness, or how often a space allows an aligner to select the best alignment.

In all cases, we report our results in terms of alignment quality, using the standard word alignment error metrics: precision, recall, F-measure and alignment error rate (Och and Ney, 2003). Our test set is the 500 manually aligned sentence pairs created by Franz Och and Hermann Ney (2003). These English-French pairs are drawn from the Canadian Hansards. English dependency trees are supplied by Minipar (Lin, 1994).

4.1 Objective Function

In our experiments, we hold all variables constant except for the alignment space being searched, and in the case of imperfect searches, the search method. In particular, all of the methods we test will use the same objective function to select the "best" alignment from their space. Let A be an alignment for an English, Foreign sentence pair, (E, F). A is represented as a set of links, where each link is a pair of English and Foreign positions, (i, j), that are connected by the alignment. The score of a proposed alignment is:

$$f_{align}(A, E, F) = \sum_{a \in A} f_{link}(a, E, F) \qquad (5)$$

Note that this objective function evaluates each link independently, unaware of the other links selected. Taskar et al (2005) have shown that with a strong f_{link}, one can achieve state of the art results using this objective function and the maximum matching algorithm. Our two experiments will vary the definition of f_{link} to test different aspects of alignment spaces.

All of the methods will create only one-to-one alignments. Phrasal alignment would introduce unnecessary complications that could mask some of the differences in the re-orderings defined by these spaces.

4.2 Search methods tested

We test seven methods, one for each of the four syntactic spaces described in this paper, and three variations of search in permutation space:

Greedy: A greedy search of permutation space. Links are added in the order of their link scores. This corresponds to the competitive linking algorithm (Melamed, 2000).

Beam: A beam search of permutation space, where links are added to a growing alignment, biased by their link scores. Beam width is 2 and agenda size is 40.

Match: The weighted maximum matching algorithm (West, 2001). This is a perfect search of permutation space.

ITG: The alignment resulting from ITG parsing with the canonical grammar in (2). This is a perfect search of ITG space.

Dep: A beam search of the dependency space. This is equivalent to **Beam** plus a dependency constraint.

D-ITG: The result of ITG parsing as described in Section 3.2. This is a perfect search of the intersection of the ITG and dependency spaces.

HD-ITG: The D-ITG method with an added head constraint, as described in Section 3.3.

4.3 Learned objective function

The link score f_{link} is usually imperfect, because it is learned from data. Appropriately defined alignment spaces may rule out bad links even if they are assigned high f_{link} values, based on other links in the alignment. We define the following simple link score to test the guidance provided by different alignment spaces:

$$f_{link}(a, E, F) = \phi^2(e_i, f_j) - C|i - j| \qquad (6)$$

Here, $a = (i, j)$ is a link and $\phi^2(e_i, f_j)$ returns the ϕ^2 correlation metric (Gale and Church, 1991)

Table 1: Results with the learned link score.

Method	Prec	Rec	F	AER
Greedy	78.1	81.4	79.5	20.47
Beam	79.1	82.7	80.7	19.32
Match	79.3	82.7	80.8	19.24
ITG	81.8	83.7	82.6	17.36
Dep	88.8	84.0	86.6	13.40
D-ITG	88.8	84.2	86.7	13.32
HD-ITG	89.2	84.0	86.9	13.15

between the English token at i and the Foreign token at j. The ϕ^2 scores were obtained using co-occurrence counts from 50k sentence pairs of Hansard data. The second term is an absolute position penalty. C is a small constant selected to be just large enough to break ties in favor of similar positions. Links to *null* are given a flat score of 0, while token pairs with no value in our ϕ^2 table are assigned -1.

The results of maximizing f_{align} on our test set are shown in Table 1. The first thing to note is that our f_{link} is not artificially weak. Our function takes into account token pairs and position, making it roughly equivalent to IBM Model 2. Our weakest method outperforms Model 2, which scores an AER of 22.0 on this test set when trained with roughly twice as many sentence pairs (Och and Ney, 2003).

The various search methods fall into three categories in terms of alignment accuracy. The searches through permutation space all have AERs of roughly 20, with the more complete searches scoring better. The **ITG** method scores an AER of 17.4, a 10% reduction in error rate from maximum matching. This indicates that the constraints established by ITG space are beneficial, even before adding an outside parse. The three dependency tree-guided methods all have AERs of around 13.3. This is a 31% improvement over maximum matching. One should also note that, with the exception of the **HD-ITG**, recall goes up as smaller spaces are searched. In a one-to-one alignment, enhancing precision can also enhance recall, as every error of commission avoided presents two new opportunities to avoid an error of omission.

The small gap between the beam search and maximum matching indicates that for this f_{link}, the beam search is a good approximation to complete enumeration of a space. This is important, as the only method we have available to search dependency space is also a beam search.

The error rates for the three dependency-based methods are similar; no one method provides much more guidance than the other. Enforcing head constraints produces only a small improvement over the **D-ITG**. Assuming our beam search is approximating a complete search, these results also indicate that D-ITG space and dependency space have very similar properties with respect to alignment.

4.4 Oracle objective function

Any time we limit an alignment space, we risk ruling out correct alignments. We now test the expressiveness of an alignment space according to the best alignments that can be found there when given an oracle link score. This is similar to the experiments in (Fox, 2002), but instead of counting crossings, we count how many links a maximal alignment misses when confined to the space.

We create a tailored f_{link} for each sentence pair, based on the gold standard alignment for that pair. Gold standard links are broken up into two categories in Och and Ney's evaluation framework (2003). S links are used when the annotators agree and are certain, while P links are meant to handle ambiguity. Since only S links are used to calculate recall, we define our f_{link} to mirror the S links in the gold standard:

$$f_{link}(a, E, F) = \begin{cases} 1 & \text{if } a \text{ is an } S \text{ in } (E, F) \\ 0 & \text{if } a \text{ is a link to } \textit{null} \\ -1 & \text{otherwise} \end{cases}$$

Table 2 shows the results of maximizing summed f_{link} values in our various alignment spaces. The two imperfect permutation searches were left out, as they are simply approximating maximum matching. The precision column was left out, as it is trivially 100 in all cases. A new column has been added to count missed links.

Maximum matching sets the upper bound for this task, with a recall of 96.4. It does not achieve perfect recall due to the one-to-one constraint. Note that its error rate is not a lower bound on the AER of a one-to-one aligner, as systems can score better by including P links.

Of the constrained systems, **ITG** fairs the best, showing only a tiny reduction in recall, due to 3 missed links throughout the entire test set. Considering the non-trivial amount of guidance provided by the **ITG** in Section 4.3, this small drop in

Table 2: Results with the perfect link score.

Method	Rec	Missed	F	AER
Dep	94.1	260	97.0	3.02
HD-ITG	94.2	258	97.0	3.00
D-ITG	94.8	232	97.3	2.69
ITG	96.3	165	98.1	1.90
Match	96.4	162	98.1	1.86

expressiveness is quite impressive. For the most part, the ITG constraints appear to rule out only incorrect alignments.

The **D-ITG** has the next highest recall, doing noticeably better than the two other dependency-based searches, but worse than the **ITG**. The 1.5% drop in expressiveness may or may not be worth the increased guidance shown in Section 4.3, depending on the task. It may be surprising to see **D-ITG** outperforming **Dep**, as the alignment space of **Dep** is larger than that of **D-ITG**. The heuristic nature of **Dep**'s search means that its alignment space is only partially explored.

The **HD-ITG** makes 26 fewer correct links than the **D-ITG**, each corresponding to a single missed link in a different sentence pair. These misses occur in cases where two modifiers switch position with respect to their head during translation. Surprisingly, there are regularly occurring, systematic constructs that violate the head constraints. An example of such a construct is when an English noun has both adjective and noun modifiers. Cases like "Canadian Wheat Board" are translated as, "Board Canadian of Wheat", switching the modifiers' relative positions. These switches correspond to discontinuous constituents (Melamed, 2003) in general bitext parsing. The **D-ITG** can handle discontinuities by freely grouping constituents to create continuity, but the **HD-ITG**, with its fixed head and modifiers, cannot. Given that the **HD-ITG** provides only slightly more guidance than the **D-ITG**, we recommend that this type of head information be included only as a soft constraint.

5 Conclusion

We have presented two new alignment spaces based on a dependency tree provided for one of the sentences in a sentence pair. We have given grammars to conduct a perfect search of these spaces using an ITG parser. The grammars derive exactly one structure for each alignment.

We have shown that syntactic constraints alone can have a very positive effect on alignment error rate. With a learned objective function, ITG constraints reduce maximum matching's error rate by 10%, while D-ITG constraints produce a 31% reduction. This gap in error rate demonstrates that a dependency tree over the English sentence can be a very powerful tool when making alignment decisions. We have also shown that while dependency constraints might limit alignment expressiveness too much for some tasks, enforcing ITG constraints results in almost no reduction in achievable recall.

References

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):263–312.

H. J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP*, pages 304–311.

W. A. Gale and K. W. Church. 1991. Identifying word correspondences in parallel texts. In *4th Speech and Natural Language Workshop*, pages 152–157. DARPA.

D. Lin and C. Cherry. 2003. Word alignment with cohesion constraint. In *HLT-NAACL 2003: Short Papers*, pages 49–51, Edmonton, Canada, May.

D. Lin. 1994. Principar - an efficient, broad-coverage, principle-based parser. In *Proceedings of COLING*, pages 42–48, Kyoto, Japan.

I. D. Melamed. 2000. Models of translational equivalence among words. *Computational Linguistics*, 26(2):221–249.

I. D. Melamed. 2003. Multitext grammars and synchronous parsers. In *HLT-NAACL 2003: Main Proceedings*, pages 158–165, Edmonton, Canada, May.

F. J. Och and H. Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–52, March.

B. Taskar, S. Lacoste-Julien, and D. Klein. 2005. A discriminative matching approach to word alignment. In *Proceedings of HLT-EMNLP*, pages 73–80, Vancouver, Canada.

D. West. 2001. *Introduction to Graph Theory*. Prentice Hall, 2nd edition.

D. Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):374.

K. Yamada and K. Knight. 2001. A syntax-based statistical translation model. In *Meeting of the Association for Computational Linguistics*, pages 523–530.

R. Zens and H. Ney. 2003. A comparative study on reordering constraints in statistical machine translation. In *Meeting of the Association for Computational Linguistics*, pages 144–151.

H. Zhang and D. Gildea. 2004. Syntax-based alignment: Supervised or unsupervised? In *Proceedings of COLING*, Geneva, Switzerland, August.