Synchronous Tree Adjoining Machine Translation

Steve DeNeefe and **Kevin Knight**

USC Information Sciences Institute

4676 Admiralty Way, Suite 1001

Marina del Rey, CA 90292 USA

{sdeneefe,knight}@isi.edu

Abstract

Tree Adjoining Grammars have well-known advantages, but are typically considered too difficult for practical systems. We demonstrate that, when done right, adjoining improves translation quality without becoming computationally intractable. Using adjoining to model optionality allows general translation patterns to be learned without the clutter of endless variations of optional material. The appropriate modifiers can later be spliced in as needed.

In this paper, we describe a novel method for learning a type of Synchronous Tree Adjoining Grammar and associated probabilities from aligned tree/string training data. We introduce a method of converting these grammars to a weakly equivalent tree transducer for decoding. Finally, we show that adjoining results in an end-to-end improvement of +0.8 Bleu over a baseline statistical syntax-based MT model on a large-scale Arabic/English MT task.

1 Introduction

Statistical MT has changed a lot in recent years. We have seen quick progress from manually crafted linguistic models to empirically learned statistical models, from word-based models to phrase-based models, and from string-based models to tree-based models. Recently there is a swing back to incorporating more linguistic information again, but this time linguistic insight carefully guides the setup of empirically learned models.

Shieber (2007) recently argued that probabilistic Synchronous Tree Adjoining Grammars (Shieber and Schabes, 1990) have the right combination of properties that satisfy both linguists and empirical MT practitioners. So far, though, most work in this area has been either more linguistic than statistical (Abeille et al., 1990) or statistically-based, but linguistically light (Nesson et al., 2006).

Current tree-based models that integrate linguistics and statistics, such as GHKM (Galley et al., 2004), are not able to generalize well from a single phrase pair. For example, from the data in Figure 1, GHKM can learn rule (a) to translate nouns with two pre-modifiers, but does not generalize to learn translation rules (b) - (d) without the optional adjective or noun modifiers. Likewise, none of these rules allow extra material to be introduced, e.g. "Pakistan's national defense minister". In large enough training data sets, we see many examples of all the common patterns, but the rarer patterns have sparse statistics or poor coverage.
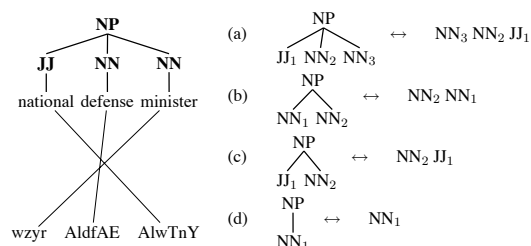

Figure 1: Rule (a) can be learned from this training example. Arguably, the more general rules (b) - (d) should also be learnable.

To mitigate this problem, the parse trees used as training data for these systems can be binarized (Wang et al., 2007). Binarization allows rules with partial constituents to be learned, resulting in more general rules, richer statistics, and better phrasal coverage (DeNeefe et al., 2007), but no principled required vs. optional decision has been made. This method's key weakness is that binarization always keeps adjacent siblings together, so there is no way to group the head with a required complement if optional information intervenes between the two. Furthermore, if all kinds of children are considered equally optional, then we have removed important syntactic constraints, which may end up permitting too much freedom. In addition, spurious alignments may limit the binarization tech-

727

nique's effectiveness.

In this paper, we present a method of learning a type of probabilistic Synchronous Tree Adjoining Grammar (STAG) automatically from a corpus of word-aligned tree/string pairs. To learn this grammar we use linguistic resources to make the required vs. optional decision. We then directly model the optionality in the translation rules by learning statistics for the required parts of the rule independently from the optional parts. We also present a method of converting these rules into a well-studied tree transducer formalism for decoding purposes. We then show that modeling optionality using adjoining results in a statistically significant BLEU gain over our baseline syntax-based model with no adjoining.

2 Translation Model

2.1 Synchronous Tree Insertion Grammars

Tree Adjoining Grammars (TAG), introduced by Joshi et al. (1975) and Joshi (1985), allow insertion of unbounded amounts of material into the structure of an existing tree using an adjunction operation. Usually they also include a substitution operation, which has a 'fill in the blank' semantics, replacing a substitution leaf node with a tree. Figure 2 visually demonstrates TAG operations. Shieber and Schabes (1990) offer a synchronous version of TAG (STAG), allowing the construction of a pair of trees in lockstep fashion using the TAG operations of substitution and adjunction on tree *pairs*. To facilitate this synchronous behavior, links between pairs of nodes in each tree pair define the possible *sites* for substitution and adjunction to happen. One application of STAG is machine translation (Abeille et al., 1990).

One negative aspect of TAG is the computational complexity: $O(n^6)$ time is required for monolingual parsing (and thus decoding), and STAG requires $O(n^{12})$ for bilingual parsing (which might be used for training the model directly on bilingual data). Tree Insertion Grammars (TIG) are a restricted form of TAG that was introduced (Schabes and Waters, 1995) to keep the same benefits as TAG (adjoining of unbounded material) without the computational complexity—TIG parsing is $O(n^3)$. This reduction is due to a limitation on adjoining: auxiliary trees can only introduce tree material to the left or the right of the node adjoined to. Thus an auxiliary tree can be classified by direction as left or right adjoining.
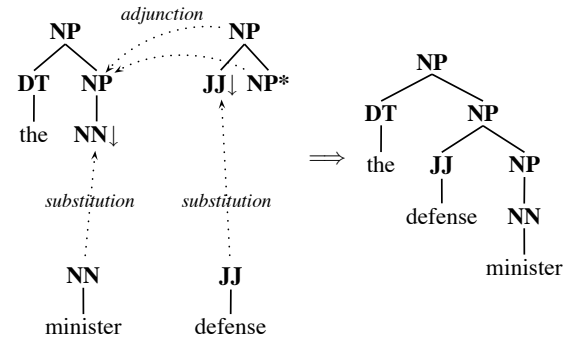

Figure 2: TAG grammars use substitution and adjunction operations to construct trees. Substitution replaces the substitution node (marked with ↓) with another tree. Adjunction inserts an auxiliary tree—a special kind of tree fragment with a foot node (marked with *)—into an existing tree at a permitted non-terminal node. Note that in TAG, adjunctions are permitted at *any* non-terminal with the same label as the root and foot node of the auxiliary tree, while in STAG adjunctions are restricted to linked sites.

Nesson et al. (2006) introduce a probabilistic, synchronous variant of TIG and demonstrate its use for machine translation, showing results that beat both word-based and phrase-based MT models on a limited-vocabulary, small-scale training and test set. Training the model uses an $O(n^6)$ bilingual parsing algorithm, and decoding is $O(n^3)$. Though this model uses trees in the formal sense, it does not create Penn Treebank (Marcus et al., 1993) style linguistic trees, but uses only one non-terminal label (**X**) to create those trees using six simple rule structures.

The grammars we use in this paper share some properties in common with those of Nesson et al. (2006) in that they are of the probabilistic, synchronous tree-insertion variety. All pairs of sites (both adjunction and substitution in our case) are explicitly linked. Adjunction sites are restricted by direction: at each linked site, the source and target side each specify one allowed direction. The result is that each synchronous adjunction site can be classified into one of four direction classes: {LR, LL, RR, RL}. For example, LR means the source side site only allows left adjoining trees and the target side site only allows right adjoining trees.

There are several important differences between our grammars and the ones of Nesson et al. (2006):

**Richer, Linguistic Trees:** Our grammars have a

Penn Treebank-style linguistic tree on the English (target) side, and a hierarchical structure using only a single non-terminal symbol (**X**) on the source side. We believe this provides the rich information needed in the target language without over-constraining the model.

Substitution Sites/Non-lexical trees: We use both substitution and adjunction (Nesson et al. (2006) only used adjunction) and do not require all trees to contain lexical items as is commonly done in TIG (Schabes and Waters, 1995).

Single Adjunction/Multiple Sites: Each non-terminal node in a tree may allow multiple adjunction sites, but every site only allows at most one adjunction,[1] a common assumption for TAG as specified in the Vijay-Shanker (1987) definition.

Here are some examples of automatically learned translation rules with interpretations of how they work:

1. simple lexical rules for translating words or phrases:

$$
\begin{array}{ccc}
\mathbf{IN} & & \mathbf{X} \\
| & \iff & | \\
\text{without} & & \text{AlA}
\end{array}
$$
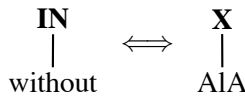
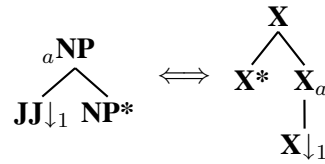   interpretation: translate the Arabic word "AlA" as the preposition "without"

2. rules with substitution for translating phrases with holes (substitution sites are designated by an arrow and numeric subscript, e.g. $\mathbf{NP}\!\downarrow_1$):
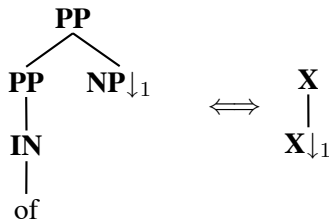
   interpretation: insert "of" to turn a noun phrase into a prepositional phrase

3. simple adjoining rules for inserting optional modifiers (adjoining sites are designated by

an alphabetic subscript before *or* after a non-terminal to indicate direction of adjoining, e.g. $_a\mathbf{NP}$):

   interpretation: adjoin an adjective before a noun in English but after in Arabic, and allowing further adjoinings in those same directions afterward

4. rules with multiple adjunction and substitution sites:
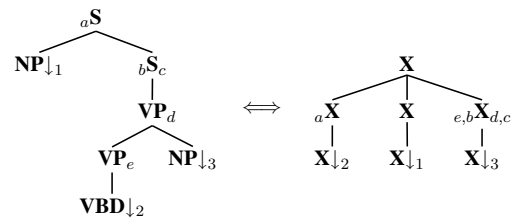
   interpretation: translate an Arabic sentence in VSO form into an English sentence in SVO form, with multiple adjoining options

2.2 Generative Story

When we use these rules to translate from a foreign sentence f into an English sentence e, we use several models together in a log-linear fashion, but our primary model is a joint model of $P(e_{tree}, f_{tree})$, which is our surrogate for directly modeling $P(e|f)$. This can be justified because $P(e|f) = \frac{P(e,f)}{P(f)}$, and $P(f)$ is fixed for a given foreign sentence. Therefore:

$$
\begin{aligned}
\operatorname*{argmax}_{e} P(e|f) &= \operatorname*{argmax}_{e} P(e,f) \\
&\approx \operatorname{yield}(\operatorname*{argmax}_{e_{tree}} P(e_{tree}, f_{tree})) \\
&\approx \operatorname{yield}(\operatorname*{argmax}_{e_{tree}} P(d_{e_{tree}, f_{tree}}))
\end{aligned}
$$

where $d_{e_{tree}, f_{tree}}$ is a derivation tree of rules that generates e_{tree} and f_{tree}. In other words, e, the highest probability translation of f, can be approximated by taking the yield of the highest probability tree e_{tree} that is a translation of the highest probability tree of f. This can further be approximated by the highest probability derivation of rules translating between f and e via trees.

Now we define the probability of generating $d_{e_{tree}, f_{tree}}$. Starting with an initial symbol pair

representing a rule with a single substitution site,[2] $\langle \text{TOP}\downarrow, \text{X}\downarrow \rangle$, a tree pair can be generated by the following steps:

1. For each substitution site s_i in the current rule r_1:

   (a) Choose with probability $P_{sub}(r_2|\langle \text{label}_L(s_i), \text{label}_R(s_i) \rangle)$ a rule r_2 having root node labels $\text{label}_L(s_i)$ and $\text{label}_R(s_i)$ that match the left and right labels at s_i.

2. For each adjunction site s_{i,r_1} in the current rule r_1:

   (a) Choose with rule-specific probability $P_{\text{ifadj}}(\text{decision}_{adjoin}|s_{i,r_1}, r_1)$ choose whether or not to adjoin at the current site s_{i,r_1}.

   (b) If we *are* adjoining at site s_{i,r_1}, choose with probability $P_{adj}(r_2|d, \langle \text{label}_L(s_{i,r_1}), \text{label}_R(s_{i,r_1}) \rangle)$ a rule r_2 of direction class d having root node labels $\text{label}_L(s_{i,r_1})$ and $\text{label}_R(s_{i,r_1})$ that match the left and right labels at s_{i,r_1}.

3. Recursively process each of the added rules

For all substitution rules r_s, adjoining rules r_a, and adjoining sites $s_{i,r}$, the probability of a derivation tree using these rules is the product of all the probabilities used in this process, i.e.:

$$
\begin{aligned}
P_{deriv} = \\
\prod_{r_s} \Bigg(P_{sub}(r_s|\langle \text{root}_L(r_s), \text{root}_R(r_s) \rangle) \cdot \\
\prod_{s_{i,r_s}} P_{\text{ifadj}}(\text{decision}_{adjoin}|s_{i,r_s}, r_s) \Bigg) \cdot \\
\prod_{r_a} \Bigg(P_{adj}(r_a|\text{dir}(r_a), \langle \text{root}_L(r_a), \text{root}_R(r_a) \rangle) \cdot \\
\prod_{s_{i,r_a}} P_{\text{ifadj}}(\text{decision}_{adjoin}|s_{i,r_a}, r_a) \Bigg)
\end{aligned}
$$

Note that while every new substitution site requires an additional rule to be added, adjunction sites may or may not introduce an additional rule based on the rule-specific P_{ifadj} probability. This allows adjunction to represent linguistic optionality.

---

[2]Here and in the following, we use *site* as shorthand for synchronous site pair.

3 Learning the Model

Instead of using bilingual parsing to directly train our model from strings as done by Nesson et al. (2006), we follow the method of Galley et al. (2004) by dividing the training process into steps. First, we word align the parallel sentences and parse the English (target) side. Then, we transform the aligned tree/string training data into derivation trees of minimal translation rules (Section 3.1). Finally, we learn our probability models $P_{sub}, P_{\text{ifadj}}$, and P_{adj} by collecting counts over the derivation trees (Section 3.2). This method is quick enough to allow us to scale our learning process to large-scale data sets.

3.1 Generating Derivation Trees and Rules

There are four steps in transforming the training data into derivation trees and rules, the first two operating only on the English parse tree itself:[3]

A. Marking Required vs. Optional. For each constituent in the English parse tree, we mark children as (H)ead, (R)equired, or (O)ptional elements (see step (a) in Figure 3). The choice of head, required, or optional has a large impact on the generality and applicability of our grammar. If all children are considered required, the result is the same as the GHKM rules of Galley et al. (2004) and has the same problem—lots of low count, syntactically over-constrained rules. Too many optional children, on the other hand, allows ungrammatical output. Our proposed model is a linguistically motivated middle ground: we consider the linguistic heads and complements selected by Collins' (2003) rules to be required and all other children to be optional.

B. Parse tree to TIG tree. Next, we restructure the English tree to form a TIG derivation where head and required elements are substitutions, and optional elements are adjunctions (see step (b) in Figure 3). To allow for adjoining between siblings under a constituent, we first do a head-out binarization of the tree. This is followed by excising[4] any children marked as optional and replacing them with an adjunction site, as shown in Figure 4. Note that we excise a chain of optional children as one site with each optional child

---

[3]These first two steps were inspired by the method Chiang (2003) used to automatically extract a TIG from an English parse tree.

[4]Excising is the opposite of adjoining: extracting out an auxiliary rule from a tree to form two smaller trees.
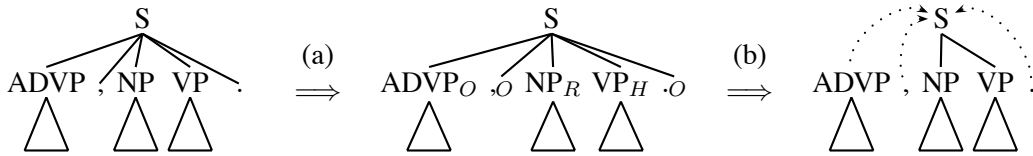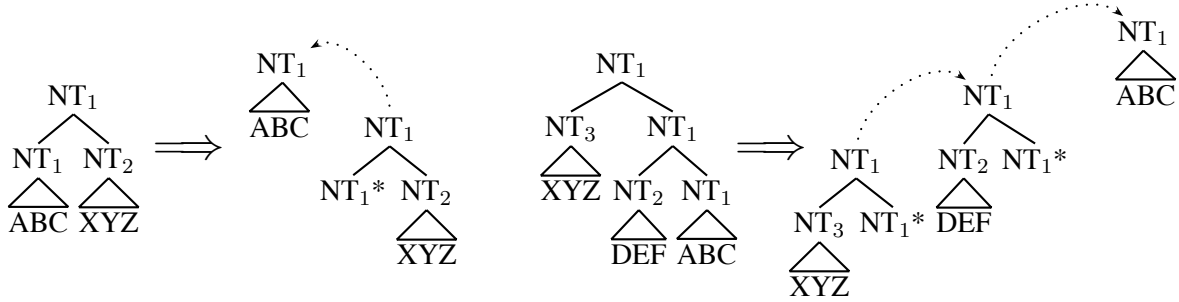
Figure 3: Parse tree to TIG transformation: (a) mark constituent children with (H)ead, (R)equired, and (O)ptional, then (b) restructure the tree so that head and required elements are substitutions, while optional elements are adjoined (shown with dotted lines).

(a) excising one optional child (XYZ) (b) excising a series of optional children (DEF, then XYZ)

Figure 4: Two examples of excising auxiliary trees from a head-out binarized parse tree: (a) excising one optional left branch, (b) excising a chain of optional branches in the same (right) direction into a series of adjunctions. In both examples, the 'ABC' child is the head, while the other children are optional.

adjoined to the previous child, as in Figure 4(b).

C. Extracting rules and derivation trees. We now have a TIG derivation tree, with each elementary tree attached to its parent by a substitution or adjunction link. We can now extract synchronous rules allowed by the alignments and syntactic constituents. This can be done using a method inspired by the rule-extraction approach of Galley et al. (2004), but instead of directly operating on the parse tree we process the English TIG derivation tree. In bottom-up fashion, we visit each elementary tree in the derivation, allowing a rule rooted at this tree to be extracted if its words or those of its descendants are aligned such that they are the English side of a self-contained parallel phrase (i.e., the foreign text of this phrase is not aligned to English leaves outside of the set of descendants). Otherwise, this elementary tree is rejoined with its parent to form a larger elementary tree. At the end of this process we have a new set of linked elementary trees which make up the English side of the grammar, where each substitution or adjunction link becomes a substitution or adjunction site in the synchronous grammar.

On the foreign side we start with the foreign text of the self-contained parallel phrase and replace any parts of this phrase covered by substituted or adjoined children of the English side tree with substitution sites or adjunction site markers. From this, we produce a tree with a simple, regular form by placing all items under a root node labeled X. In the case of more than one foreign word or substitution site, we introduce an intermediate level of X-labeled non-terminals to allow for possible adjunction between elements, otherwise the adjoining sites attach to the single root node. We attach all foreign-side adjoining sites to be left adjoining, except on the right side of the right-hand child.

It is possible to have the head child tree on the English side not aligned to anything, while the adjoined children are. This may lead to rules with no foreign non-terminal from which to anchor the adjunctions, so in this case, we attach adjoined child elementary trees starting from the head and moving out until we attach a some child with a non-empty foreign side.

D. Generalizing rules. We need to clarify what makes one rule distinct from another. Consider the example in Figure 5, which shows selected rules learned in the case of two different noun phrases. If the noun phrase consists of just a single noun, we learn rule (a), while if the noun phrase also has an adjective, we learn rules (b) and (c). Since adjoining the adjective is optional, *we*

731

consider rules (a) and (c) to be the same rule, the latter with an adjoining seen, and the former with the same adjoining not seen.

3.2 Statistical Models

Once we have the derivation trees and list of rules, we learn our statistical models using maximum likelihood estimation. By counting and normalizing appropriately over the entire corpus, we can straightforwardly learn the P_{sub} and P_{adj} distributions. However, recall that in our model P_{ifadj} is a rule-specific probability, which makes it more difficult to estimate accurately. For common rules, we see plenty of examples of adjoining, while for other rules, we need to learn from only a handful of examples. Smoothing and generalization are especially important for these low frequency cases.

Two options present themselves for how to estimate adjoining:

(a) A joint model of adjoining. We assume that adjoining decisions are made in combination with each other, and so learn non-zero probabilities only for adjoining combinations seen in data

(b) An independent model of adjoining. We assume adjoining decisions are made independently, and learn a model for each adjoining site separately

Option (a) may be sufficient for frequent rules, and will accurately model dependencies between different kinds of adjoining. However, it does not allow us to generalize to unseen patterns of adjoining. Consider the low frequency situation depicted in Figure 6, rules (d)-(f). We may have seen this rule four times, once with adjoining site a, twice with adjoining sites a and b, and once with a third adjoining site c. The joint model will give a zero probability to unseen patterns of adjoining, e.g. no adjoining at any site or adjoining at site b alone. Even if we use a discounting method to give a non-zero probability to unseen cases, we still have no way to distinguish one from another.

Option (b) allows us to learn reasonable estimates for these missing cases by separating out adjoining decisions and letting each speak for itself. To properly learn non-zero probabilities for unseen cases[5] we use add k smoothing ($k = \frac{1}{2}$).

A weakness of this approach still remains: adjoining is not a truly independent process, as we observe empirically in the data. In real data, frequent rules have many different observed adjoining sites (10 or 20 in some cases), many of which represent already infrequent sites in combinations never seen together. To reduce the number of invalid combinations produced, we only allow adjoinings to be used at the same time if they have occurred together in the training data. This restriction makes it possible to do less adjoining than observed, but not more. For the example in Figure 6, in addition to the observed patterns, we would also allow site b to be used alone, and we would allow no adjoinings, but we would not allow combinations of site c with either a or b. Later, we will see that this makes the decoding process more efficient.

Because both option (a) and (b) above have strengths and weaknesses, we also explore a third option which builds upon the strengths of each:

(c) A log-linear combination of the joint model and independent model. We assume the probability has both a dependent and independent element, and learn the relative weight between them automatically

To help smooth this model we add two additional binary features: one indicating adjoining patterns seen in data and one indicating previously unseen patterns.

4 Decoding

To translate with these rules, we do a monolingual parse using the foreign side of the rules (constraining the search using non-terminal labels from both sides), while keeping track of the English side string and structure for language modeling purposes. This produces all valid derivations of rules whose foreign side yield is the input string, from which we simply choose the one with the highest log-linear model score. Though this process could be done directly using a specialized parsing algorithm, we note that these rules have weakly equivalent counterparts in the Synchronous Tree Substitution Grammar (STSG) and Tree-to-string transducer (xLNTs[6]) worlds, such that each STIG rule can be translated into one equivalent rule, plus some helper rules to model the adjoin/no-adjoin

---

[5] For example, low frequency rules may have always been observed with a single adjoining pattern, and never without adjoining.

[6] xLNTs is shorthand for extended linear non-deleting top-down tree-to-string transducer.
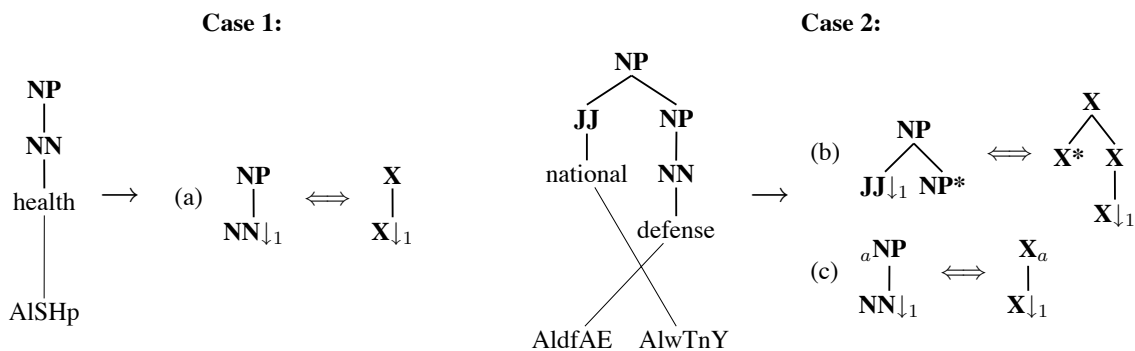
Figure 5: Selected rules learned in two cases. Rule (a) and (c) are considered the same rule, where (c) has the optional synchronous adjoining site marked with a. From these (limited) examples alone we would infer that adjective adjoining happens half the time, and is positioned before the noun in English, but after the noun in Arabic (thus the positioning of site a).
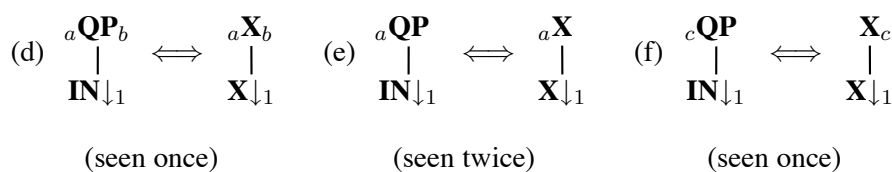

Figure 6: For a low frequency rule, we may see only a few different adjoining patterns, but we want to infer more.

decision. Conversion to a better known and explored formalism allows us to take advantage of existing code and algorithms. Here we describe the conversion process to xLNTs rules, though conversion to STSG is similar.

Algorithm 1 describes the process of converting one of our automatically learned STIG rules. On each side of the rule, we traverse the tree in a top-down, left-to-right order, recording words, substitution sites, and adjoining sites in the order encountered (left adjoinings before the node's children and right adjoinings after). We make these words and sites as the children under a single root node. The substitution sites are given states made up of a combination of their source and target labels as are the roots of non-adjoining rules. Adjoining sites are labeled with a combination of the rule id and a site id. Adjoining rule roots are labeled with a combination of the source and target root labels and the direction class. To allow for the adjoining/no-adjoining decision, two helper rules are created for each adjoining site, their root state a combination of the rule and site ids. One of these rules has only epsilon leaf nodes (representing no adjoining), while the other has leaf nodes and a state that match with the corresponding adjoining rule root (labeled with the site's source and target labels and the direction class).

For each rule, the algorithm generates one main rule and pairs of helper rules to facilitate adjoining/non-adjoining. For computational efficiency reasons, our decoder supports neither epsilon rules nor non-binary rules. So we remove epsilons using an exponential expansion of the rules: combine each main rule with an adjoining or non-adjoining helper rule for each adjunction site, then remove epsilon-only branches. For k adjunction sites this could possibly results in 2^k rules. But as discussed previously (at the end of Section 3.2), we only allow subsets of adjoining combinations seen in training data, so this number is substantially lower for large values of k.

5 Experiments

All experiments are trained with a subset (171,000 sentences or 4 million words) of the Arabic-English training data from the constrained data track of the NIST 2008 MT Evaluation, leaving out LDC2004T18, LDC2007E07, and the UN data. The training data is aligned using the LEAF technique (Fraser and Marcu, 2007). The English side of the training data is parsed with an implementation of Collins Model 2 (Collins, 2003) then head-out binarized. The tuning data (1,178 sentences) and devtest data (1,298 sentences) are

Input: Synchronous TIG rule r with j adjoining sites, $S \leftrightarrow T$, where S and T are trees
Output: a weakly equivalent xLNTs rule $S' \leftrightarrow t_1 \ldots t_n$, where S' is a one-level tree, and $2 \cdot j$
         helper rules for adjoining
Run time: $O(|S| + |T|)$
begin
  rules $\leftarrow \{\}$, lhs-state $\leftarrow$ concat('q', get-root(S), get-root(T))
  site-and-word-list-s $\leftarrow$ get-sites-and-words-in-order(S)
  site-and-word-list-t $\leftarrow$ get-sites-and-words-in-order(T)
  if r *is adjoining* **then** lhs-state $\leftarrow$ concat(lhs-state, get-adjoin-dir(S), get-adjoin-dir(T))
  lhs $\leftarrow$ construct-LHS(lhs-state, get-root(S), site-and-word-list-s)
  rhs $\leftarrow$ construct-RHS(add-states(id(r), site-and-word-list-t))
  add(rules, 'lhs $\leftrightarrow$ rhs')   `/* main rule */`
  foreach *adjoining site* $i \in 1 \ldots k$ **do**
    lhs-state $\leftarrow$ concat('q', id(r), i), rhs-state $\leftarrow$ concat('q', lhs-root)
    lhs-root $\leftarrow$ concat(source-label(i), target-label(i), source-dir(i), target-dir(i))
    lhs $\leftarrow$ construct-LHS(lhs-state, lhs-root, lhs-root)
    rhs $\leftarrow$ construct-RHS($\{$(rhs-state, lhs-root)$\}$)
    rhs-eps $\leftarrow$ construct-RHS(ϵ)
    add(rules, $\{$'lhs $\leftrightarrow$ rhs', 'lhs $\leftrightarrow$ rhs-eps'$\}$)   `/* helper rules for site i */`
  return rules
end
function *get-sites-and-words-in-order(node)*
  $y \leftarrow \{\}$
  if *node is substitution site or word* **then** append site or word to y **else**
    append left adjoining sites to y in outside-to-inside order
    foreach *child c of node* **do** append result of get-yield(c) to y
    append right adjoining sites to y in inside-to-outside order
  return y
end
function *add-states(ruld-id, node-list)*
  foreach *substitution or adjunction site s_i and in node-list* **do**
    if s_i *is substitution site* **then** state = concat('q', source-site-label(s_i), target-site-label(s_i))
    else state = concat('q', rule-id, i)
    replace s_i with (state, s_i)
  return modified node-list
end

Algorithm 1: Conversion from synchronous TIG rules to weakly equivalent xLNTs rules

	description	BLEU DevTest	NIST06
(1)	baseline: all required (GHKM minimal, head-out binarized parse trees)	48.0	47.0
(2)	joint adjoining prob model alone (only observed adjoining patterns)	48.0	46.6
(3)	independent adjoining prob model alone (only observed adjoining patterns)	48.1	46.7
(4)	independent adjoining prob model alone (with new adjoining patterns)	48.5	47.6
(5)	independent model alone + features (adjoining pattern, direction)	48.4	47.7
(6)	log-linear combination of joint & independent models + features	48.7	47.8

Table 1: End-to-end MT results show that the best adjoining model using a log-linear combination of joint and independent models (line 6) outperforms the baseline (line 1) by +0.7 and +0.8 BLEU, a statistically significant difference at the 95% confidence level.

made up of newswire documents drawn from the NIST MT evaluation data from 2004, 2005, and 2006 (GALE part). We use the newswire documents from the NIST part of the 2006 evaluation data (765 sentences) as a held-out test set.

We train our feature weights using max-BLEU (Och, 2003) and decode with a CKY-based decoder that supports language model scoring directly integrated into the search.

In addition to P_{sub}, P_{adj}, and P_{ifadj}, we use several other features in our log-linear model during decoding, including: lexical and phrase-based translation probabilities, a model similar to conditional probability on the trees $(P(f_{tree}(rule)|e_{tree}(rule)))$, a probability model for generating the top tree non-terminal, a 5-gram language model[7], and target length bonus. We also have several binary features—lexical rule, rule with missing or spurious content words—and several binary indicator features for specialized rules: unknown word rules; name, number, and date translation rules; and special fail-safe monotone translation rules in case of parse failures and extremely long sentences.

Table 1 shows the comparison between our baseline model (minimal GHKM on head-out binarized parse trees) and different models of adjoining, measured with case-insensitive, NIST-tokenized BLEU (IBM definition). The top section (lines 1–4) compares the joint adjoining probability model to the independent adjoining probability model and seen vs. unseen adjoining combinations. While the joint model results in a BLEU score at the same level as our baseline (line 2), the independent model (line 4) improves BLEU by +0.5 and +0.6, which are significant differences at the 95% confidence level. Since with the independent model we introduce both new adjoining patterns and a different probability model for adjoining (each site is independent), we also use the independent model with only previously seen adjoining patterns (line 3). The insignificant difference in BLEU between lines 2 and 3 leads us to think that the new adjoining patterns are where the improvement comes from, rather than the independent probability model alone.

We also test several other features and combinations. First, we add binary features to indicate a new adjoining combination vs. one previously seen in data. We also add features to indicate the direction class of adjoining to test if there is a systematic bias toward particular directions. These features cause no significant difference in score (line 5). We also add the joint-adjoining probability as a feature, allowing it to be combined in a log-linear fashion with the independent probability (line 6). This results in our best BLEU gain: +0.7 and +0.8 over our non-adjoining baseline.

6 Conclusion

We have presented a novel method for learning the rules and probabilities for a new statistical, linguistically-informed, syntax-based MT model that allows for adjoining. We have described a method to translate using this model. And we have demonstrated that linguistically-motivated adjoining improves the end-to-end MT results.

There are many potential directions for research to proceed. One possibility is to investigate other methods of making the required vs. optional decision, either using linguistic resources such as COMLEX or automatically learning the distinction using EM (as done for tree binarization by Wang et al. (2007)). In addition, most ideas presented here are extendable to rules with linguistic trees on both sides (using insights from Lavie et al. (2008)). Also worth investigating is the direct integration of bilingual dictionaries into the grammar (as suggested by Shieber (2007)). Lastly, rule composition and different amounts of lexicalization (Galley et al., 2006; Marcu et al., 2006; DeNeefe et al., 2007) or context modeling (Mariño et al., 2006) have been successful with other models.

Acknowledgments

References

Anne Abeille, Yves Schabes, and Aravind K. Joshi. 1990. Using lexicalized TAGs for machine translation. In *Proc. COLING*, volume 3.

---

[7]The 5-gram LM was trained on 2 billion words of automatically selected collections taken from the NIST 08 allowable data.

David Chiang. 2003. Statistical parsing with an automatically extracted tree adjoining grammar. *Data-Oriented Parsing*.

Michael Collins. 2003. Head-driven statistical models for natural language parsing. *Computational Linguistics*, 29(4).

Steve DeNeefe, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proc. EMNLP-CoNLL*.

Alexander Fraser and Daniel Marcu. 2007. Getting the structure right for word alignment: LEAF. In *Proc. EMNLP-CoNLL*.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proc. HLT-NAACL*.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proc. ACL*.

Aravind K. Joshi, L. S. Levy, and M. Takahashi. 1975. Tree adjunct grammars. *Journal of Computer and System Sciences*, 10(1).

Aravind K. Joshi. 1985. How much context-sensitivity is necessary for characterizing structural descriptions—tree adjoining grammars. *Natural Language Processing—Theoretical, Computational, and Psychological Perspectives*.

Alon Lavie, Alok Parlikar, and Vamshi Ambati. 2008. Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora. In *Proc. SSST*.

Daniel Marcu, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. EMNLP*.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2).

José B. Mariño, Rafael E. Banchs, Josep M. Crego, Adrià de Gispert, Patrik Lambert, José A. R. Fonollosa, and Marta R. Costa-jussà. 2006. N-gram-based machine translation. *Computational Linguistics*, 32(4).

Rebecca Nesson, Stuart M. Shieber, and Alexander Rush. 2006. Induction of probabilistic synchronous tree-insertion grammars for machine translation. In *Proc. AMTA*.

Franz Josef Och. 2003. Minimum error rate training in statistical machine translation. In *Proc. ACL*.

Yves Schabes and Richard C. Waters. 1995. Tree insertion grammar: A cubic-time, parsable formalism that lexicalizes context-free grammar without changing the trees produced. *Computational Linguistics*, 21(4).

Stuart M. Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. COLING*.

Stuart M. Shieber. 2007. Probabilistic synchronous tree-adjoining grammars for machine translation: The argument from bilingual dictionaries. In *Proc. SSST Wkshp., NAACL-HLT*.

Kumar Vijay-Shanker. 1987. *A study of tree adjoining grammars*. Ph.D. thesis.

Wei Wang, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proc. EMNLP and CoNLL*.