String-to-Dependency Statistical Machine Translation

Libin Shen[*]
Raytheon BBN Technologies

Jinxi Xu[**]
Raytheon BBN Technologies

Ralph Weischedel[†]
Raytheon BBN Technologies

We propose a novel string-to-dependency algorithm for statistical machine translation. This algorithm employs a target dependency language model during decoding to exploit long distance word relations, which cannot be modeled with a traditional n*-gram language model. Experiments show that the algorithm achieves significant improvement in MT performance over a state-of-the-art hierarchical string-to-string system on NIST MT06 and MT08 newswire evaluation sets.*

1. Introduction

n-gram Language Models (LMs) have been widely used in current Statistical Machine Translation (SMT) systems. Because they treat a sentence as a flat string of tokens, a drawback of traditional *n*-gram LMs is that they cannot model long range word relations, such as predicate–argument attachments, that are critical to translation quality.

We propose a hierarchical string-to-dependency translation model that exploits a dependency LM while decoding (as opposed to during reranking *n*-best output) to score alternative translations based on their structural soundness. In order to generate the structured output (dependency trees) required for dependency LM scoring, translation rules in our system represent the target side as dependency structures. We restrict the target side of the rules to well-formed dependency structures to weed out bad translation rules and enable efficient decoding through dynamic programming. Due to the flexibility of well-formed dependency structures, such structures can cover a large set of non-constituent transfer rules (Marcu et al. 2006) that have been shown useful for MT.

For comparison purposes, as our baseline, we replicated the Hiero decoder (Chiang 2005), a state-of-the-art hierarchical string-to-string model. Our experiments show that the string-to-dependency decoder significantly improves MT performance. Overall, the

* 10 Moulton Street, Cambridge, MA 02138. E-mail: `libinshen@gmail.com`.
** 10 Moulton Street, Cambridge, MA 02138. E-mail: `jxu@bbn.com`.
† 10 Moulton Street, Cambridge, MA 02138. E-mail: `weisched@bbn.com`.

Submission received: 6 March 2009; revised submission received: 1 December 2009; accepted for publication: 18 March 2010.

improvement in BLEU score is around 2 BLEU points on NIST Arabic-to-English and Chinese-to-English newswire test sets.

Section 2 briefly discusses previous approaches to SMT in order to motivate our work. Section 3 provides an overview of our string-to-dependency translation system. Section 4 provides a complete description of our system, including formal definitions of well-formed dependency structures and their operations, as well as proofs about their key properties. Section 5 describes the implementation details, which include rule extraction, decoding, using dependency LM scores, and using labels in translation rules. We discuss experimental results in Section 6, compare our work with related work in Section 7, and draw conclusions in Section 8.

2. Previous Approaches to SMT

Phrase-based systems (Koehn, Och, and Marcu 2003; Och 2003) had dominated SMT until recently. Such systems typically treat the input as a sequence of phrases (word *n*-grams), reorder them, and produce a translation for the reordered sentence based on translation options of each source phrase. A prominent feature of such systems is the use of an *n*-gram LM to measure the quality of translation hypotheses. A drawback of such systems is that the lack of structural information in the output makes it impossible to score translation hypotheses based on their structural soundness.

The Hiero system (Chiang 2007) was a major breakthrough in SMT. Translation rules in Hiero contain non-terminals (NTs), as well as words, which allow the input to be translated in a hierarchical manner. Because both the source and target sides of its translation rules are strings with NTs, Hiero can be viewed as a hierarchical string-to-string model. Despite the hierarchical nature of its decoder, Hiero lacks the ability to measure translation quality based on structural relations such as predicate–argument agreement.

Yamada and Knight (2001) proposed a syntax-based translation model that transfers a source parse tree into a target string. This method depends on the quality of source side parsing, and ignores target information during source side analysis. Mi, Huang, and Liu (2008) later proposed a translation model that takes the source parse forest as MT input to reduce translation errors due to imperfect source side analysis.

Galley et al. (2004) proposed an MT model which produces target parse trees for string inputs in order to exploit the syntactic structure of the target language. Galley et al. (2006) formalized this approach with tree transducers (Graehl and Knight 2004) by using context-free parse trees to represent the target side. However, it was later shown by Marcu et al. (2006) and Wang, Knight, and Marcu (2007) that coverage could be a big issue for the constituent based rules, even though the translation rule set was already very large.

Carreras and Collins (2009) introduced a string-to-tree MT model based on spinal Tree Adjoining Grammar (TAG) (Joshi and Schabes 1997; Shen, Champollion, and Joshi 2008). In this model, a translation rule is composed of a source string and a target elementary tree. Target hypothesis trees are combined with the adjoining operation, and there are no NT slots for substitution as in LTAG-spinal parsing (Shen and Joshi 2005, 2008; Carreras, Collins, and Koo 2008). Without the constraint of NT slots, the adjoining operation allows very flexible composition, so that the search space becomes much larger. One has to carefully prune the search space.

DeNeefe and Knight (2009) proposed another TAG-based MT model. In their implementation, a TAG grammar was transformed to an equivalent Tree Insertion

Grammar (TIG). In this way, they do not have an explicit adjoining operation in their system, and as such reduce the search space in decoding. Sub-trees are combined with NT substitution.

Many researchers followed the tree-to-tree approach (Shieber and Schabes 1990) to take advantage of structural knowledge on both sides—for example, as in the papers by Hajič et al. (2002), Eisner (2003), Ding and Palmer (2005), and Quirk, Menezes, and Cherry (2005). Although tree-to-tree models can represent rich structural information of the input and the output, they have not significantly improved MT performance, possibly due to a much larger grammar and search space. On the other hand, Smith and Eisner (2006) showed the necessity of allowing loose transformations between the trees, which made tree-to-tree models even more complicated.

3. Overview of String-to-Dependency Translation

Our system is designed to address problems with existing SMT approaches. It is novel in two respects. First, it uses a dependency LM to model long-distance relations. Second, it uses well-formed dependency structures to represent translation hypotheses to achieve an effective trade-off between model coverage and decoding complexity.

3.1 Dependency-Based Translation and Language Models

Our system generates target dependency trees as output and exploits a dependency LM in scoring translation hypotheses. As described before, the goal of using a dependency LM is to exploit long-distance word dependencies and as such model the quality of the output more accurately.

Figure 1 shows an example dependency tree. Each arrow points from a child to its parent. In this example, the word *find* is the root.

For the purpose of comparison, we first show how a simplified SMT system uses an n-gram LM to score translation hypotheses:

$$S'_e = \underset{S_e}{\text{argmax}}\, P(S_e|S_f)^{w_1} P(S_f|S_e)^{w_2} P(S_e)^{w_3} \tag{1}$$

where w_1, w_2, and w_3 are feature weights. S_f is the input and S_e's are outputs. $P(S_e|S_f)$ is the probability of the target string given the source, and $P(S_f|S_s)$ is the probability of the source given the target. $P(S_e)$ is the prior probability of the target string S_e using an n-gram LM.

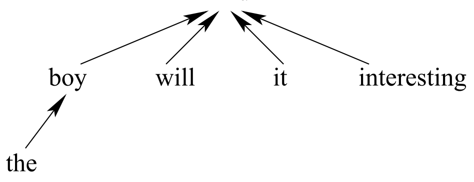

Figure 1
The dependency tree for sentence *the boy will find it interesting.*

In comparison, the scoring function in our system is:

$$D' = \operatorname*{argmax}_{D} P(D|S_f)^{w_1} P(S_f|D)^{w_2} P(D)^{w_3} \tag{2}$$

where $P(D)$ is the dependency LM score of target dependency tree D. We will show how to compute $P(D)$ in Section 5.4.

We can rewrite Equation (2) with a linear model:

$$D' = \operatorname*{argmax}_{D} \sum_{i=1}^{n} w_i F_i(S_f, D) \tag{3}$$

where $n = 3, F_1 = \log P(D|S_f), F_2 = \log P(S_f|D)$, and $F_3 = \log P(D)$. In practice, we use both a dependency LM and a traditional n-gram LM (also known as a string LM), as well as several other features, in our decoder. Section 5.6 lists all the features used in our decoder.

3.2 Well-Formed Dependency Structures

A central question in our system design is: What kinds of dependency structures are allowed in translation rules? One extreme would be to allow any arbitrary multiple level treelets, as in Ding and Palmer (2005) and Quirk, Menezes, and Cherry (2005). One can define translation rules on any fragment of a parse/dependency tree. It offers maximum coverage of translation patterns, but suffers from data sparseness and a large search space.

The other extreme would be to allow only complete (CFG) constituents. This offers a more robust model and a small search space, but excludes many useful transfer rules.

In our system, the target side hypotheses are restricted to well-formed dependency structures (see Section 4 for formal definitions) for a trade-off between rule coverage, model robustness, and decoding complexity. In short, a well-formed dependency structure is either (1) a single rooted tree, with each child being a complete sub-tree, or (2) a sequence of siblings, each being a complete sub-tree.

Well-formed dependency structures are very flexible and can represent a variety of non-constituent rules in addition to rules that are complete constituents. For example, the following translation

$$\text{hong} \xrightarrow{\textit{Chinese-to-English}} \text{the red}$$

is obviously useful for Chinese-to-English MT, but cannot be represented in some tree-based translation systems since *the red* is a partial constituent. However, it is a valid dependency structure in our system.

4. Formalism

We first formally define the well-formed dependency structures, which are used to represent target hypotheses. Then, we define the operations to build well-formed dependency structures from the bottom up in decoding.

4.1 Well-Formed Dependency Structures and Categories

In order to exclude undesirable structures and reduce the search space, we only allow S_e whose dependency structure D is well formed, which we will define subsequently. The well-formedness requirement will be applied to partial decoding results.

Based on the results of previous work (DeNeefe et al. 2007), we keep two kinds of dependency structures, fixed and floating. **Fixed structures** consist of a sub-root with children, each of which must be a complete constituent. We call them fixed dependency structures because the head is known or fixed. **Floating structures** consist of a number of consecutive sibling nodes of a common head, but the head itself is unspecified. Each of the siblings must be a complete constituent. Floating structures can represent many linguistically meaningful non-constituent structures: for example, like *the red*, a modifier of a noun. Only those two kinds of dependency structures are well-formed structures in our system.

In the rest of this section, we will provide formal definitions of well-formed structures and combinatory operations over them, so that we can easily manipulate them in decoding. Examples will be provided along with the formal definitions to aid understanding.

Consider a sentence $S = w_1 w_2 ... w_n$. Let $d_1 d_2 ... d_n$ represent the parent word IDs for each word. For example, $d_4 = 2$ means that w_4 depends on w_2. If w_i is a root, we define $d_i = 0$.

Definition 1
A dependency structure $d_i d_{i+1} ... d_j$, or $d_{i..j}$ for short, is **fixed on head** h, where $h \in [i,j]$, or **fixed** for short, if and only if it meets the following conditions

1. $d_h \notin [i,j]$
2. $\forall k \in [i,j]$ and $k \neq h$, $d_k \in [i,j]$
3. $\forall k \notin [i,j]$, $d_k = h$ or $d_k \notin [i,j]$

We say the category of $d_{i..j}$ is $(-,h,-)$, where $-$ means this field is undefined.

Definition 2
A dependency structure $d_i ... d_j$ is **floating with children** C, for a non-empty set $C \subseteq \{i, ..., j\}$, or **floating** for short, if and only if it meets the following conditions

1. $\exists h \notin [i,j], s.t. \forall k \in C, d_k = h$
2. $\forall k \in [i,j]$ and $k \notin C, d_k \in [i,j]$
3. $\forall k \notin [i,j], d_k \notin [i,j]$

We say the category of $d_{i..j}$ is $(C,-,-)$ if $j < h$, which means that children are on the left side of the head, or $(-,-,C)$ otherwise.

A **category** is composed of the three fields (A,h,B), where h is used to represent the head, and fields A and B represent left and right dependents of the head, respectively. A dependency structure is **well-formed** if and only if it is either fixed or floating.

Examples

We represent dependency structures with graphs. Figure 2 shows examples of fixed structures, Figure 3 shows examples of floating structures, and Figure 4 shows ill-formed dependency structures.

The structures in Figures 2 and 3 are well-formed. Figure 4(a) is ill-formed because *boy* does not have its child word *the* in the tree. Figure 4(b) is ill-formed because it is not a continuous segment due to the missing *it*.

As for the example *the red* mentioned earlier, it is a well-formed floating dependency structure.

It is easy to see that a floating structure whose child set C has only one element is also a fixed structure. Actually, this is a desirable property on which we will introduce meta category operations later. However, for the sake of convenience, we would like to assign a single category to each well-formed structure.

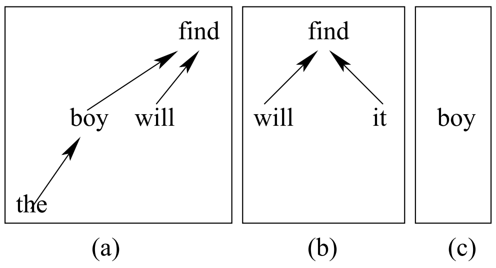

Figure 2
Fixed dependency structures.

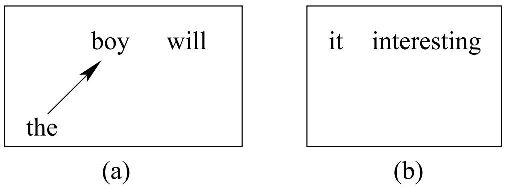

Figure 3
Floating dependency structures.

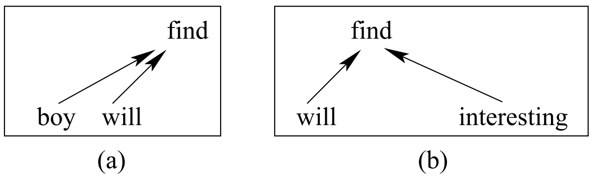

Figure 4
Ill-formed dependency structures.

Definition 3
Let structure T be well formed. Category *cat* of T is defined as follows

$$\mathrm{cat}(T) = \begin{cases} (-, h, -) & \text{if } T \text{ is fixed on } h \\ (C, -, -) & \text{if } T \text{ is floating with children } C \text{ on the left side AND } |C| > 1 \\ (-, -, C) & \text{if } T \text{ is floating with children } C \text{ on the right side AND } |C| > 1 \end{cases}$$

cat is well-defined according to Definitions 1 and 2.

4.2 Operations

One of the purposes of introducing floating dependency structures is that siblings having a common parent will become a well-defined entity, although they are not considered a constituent. We always build well-formed partial structures on the target side in decoding. Furthermore, we combine partial dependency structures in a way such that we can obtain all possible well-formed dependency structures (but no ill-formed ones) during bottom–up decoding.

The solution is to employ categories (introduced earlier). Each well-formed dependency structure has a category. We can apply four combinatory operations over the categories. If we can combine two categories with a certain category operation, we can use a corresponding tree operation to combine two dependency structures. The category of the combined dependency structure is the result of the combinatory category operations.

Operations on Well-Formed Dependency Structures

There are four types of operations on well-formed dependency structures. Instead of providing formal definitions, we use figures to illustrate these operations to make them easy to understand. Figure 1 shows a traditional dependency tree. Figure 5 shows the four operations for combining partial dependency structures, which are **left adjoining** (LA), **right adjoining** (RA), **left concatenation** (LC), and **right concatenation** (RC). We always combine two well-formed structures in one of the four ways, and obtain a larger well-formed structure.

Two structures can be combined by adjoining, which is similar to the traditional dependency formalism. We can adjoin either a fixed structure or a floating structure to the head of a fixed structure.

Completed siblings can be combined via concatenation. We can concatenate two fixed structures, one fixed structure with one floating structure, or two floating structures in the same direction.

The flexibility of the order of operation allows us to take advantage of various translation fragments encoded in transfer rules. Figure 6 shows alternative ways of applying operations on well-formed structures to build larger structures in a bottom–up style. Numbers represent the order of operation. The fact that the same dependency structure can have multiple derivations means that we can utilize various rules learned from different training samples. Such flexibility is important for MT.

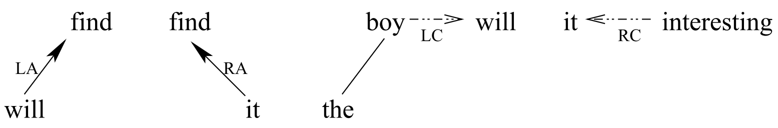

Figure 5
Operations over well-formed structures.

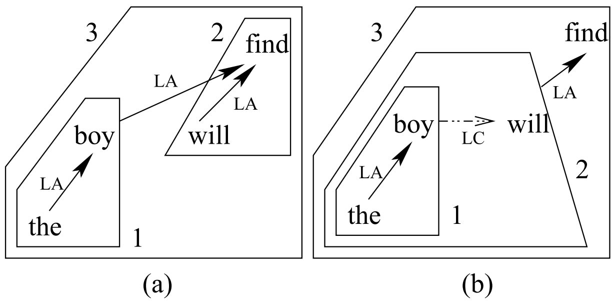

Figure 6
Two alternative derivations of an example dependency tree.

Meta Operations on Categories

We first introduce three meta category operations, which will later be used to define category operations. Two of the meta operations are unary operations, **left raising** (LR) and **right raising** (RR), and one is the binary operation **unification** (UF).

Definition 4
Meta Category Operations

- $\mathrm{LR}((-,h,-)) = (\{h\},-,-)$
- $\mathrm{RR}((-,h,-)) = (-,-,\{h\})$
- $\mathrm{UF}((A_1,h_1,B_1),(A_2,h_2,B_2)) = \mathrm{NORM}((A_1 \sqcup A_2, h_1 \sqcup h_2, B_1 \sqcup B_2))$

First, the raising operations are used to turn a completed fixed structure into a floating structure, according to Theorem 1.

Theorem 1
A fixed structure with category $(-,h,-)$ for span $[i,j]$ is also a floating structure with children $\{h\}$ if there are no outside words depending on word h, which means that

$$\forall k \notin [i,j], d_k \neq h \tag{4}$$

Proof
It suffices to show that all the three conditions of floating structures hold. Conditions 1 and 2 immediately follow from conditions 1 and 2 of the fixed structure, respectively. Condition 3 is met according to Equation (4) and condition 3 of the fixed structure. ■

Therefore, we can always raise a fixed structure if we assume it is complete, that is, Equation (4) holds.

Unification is well-defined if and only if we can unify all three elements and the result is a valid fixed or floating category. For example, we can unify a fixed structure with a floating structure or two floating structures in the same direction, but we cannot unify two fixed structures.

$$h_1 \sqcup h_2 = \begin{cases} h_1 & \text{if } h_2 = - \\ h_2 & \text{if } h_1 = - \\ \text{undefined} & \text{otherwise} \end{cases}$$

$$A_1 \sqcup A_2 = \begin{cases} A_1 & \text{if } A_2 = - \\ A_2 & \text{if } A_1 = - \\ A_1 \cup A_2 & \text{otherwise} \end{cases}$$

$$\text{NORM}((A, h, B)) = \begin{cases} (-, h, -) & \text{if } h \neq - \\ (A, -, -) & \text{if } h = -, B = - \\ (-, -, B) & \text{if } h = -, A = - \\ \text{undefined} & \text{otherwise} \end{cases}$$

Operations on Categories

Now we define category operations. For the sake of convenience, we use the same names for category operations and dependency structure operations. We can easily use the meta category operations to define the four combinatory category operations. The definition of the operations is as follows.

Definition 5
Combinatory category operations

$$\begin{aligned}
\text{LA}((A_1,-,-),(-,h_2,-)) &= \text{UF}((A_1,-,-),(-,h_2,-)) \\
\text{LA}((-,h_1,-),(-,h_2,-)) &= \text{UF}(\text{LR}((-,h_1,-)),(-,h_2,-)) \\
\text{LC}((A_1,-,-),(A_2,-,-)) &= \text{UF}((A_1,-,-),(A_2,-,-)) \\
\text{LC}((A_1,-,-),(-,h_2,-)) &= \text{UF}((A_1,-,-),\text{LR}((-,h_2,-))) \\
\text{LC}((-,h_1,-),(A_2,-,-)) &= \text{UF}(\text{LR}((-,h_1,-)),(A_2,-,-)) \\
\text{LC}((-,h_1,-),(-,h_2,-)) &= \text{UF}(\text{LR}((-,h_1,-)),\text{LR}((-,h_2,-))) \\
\text{RA}((-,h_1,-),(-,-,B_2)) &= \text{UF}((-,h_1,-),(-,-,B_2)) \\
\text{RA}((-,h_1,-),(-,h_2,-)) &= \text{UF}((-,h_1,-),\text{RR}((-,h_2,-))) \\
\text{RC}((-,-,B_1),(-,-,B_2)) &= \text{UF}((-,-,B_1),(-,-,B_2)) \\
\text{RC}((-,h_1,-),(-,-,B_2)) &= \text{UF}(\text{RR}((-,h_1,-)),(-,-,B_2)) \\
\text{RC}((-,-,B_1),(-,h_2,-)) &= \text{UF}((-,-,B_1),\text{RR}((-,h_2,-))) \\
\text{RC}((-,h_1,-),(-,h_2,-)) &= \text{UF}(\text{RR}((-,h_1,-)),\text{RR}((-,h_2,-)))
\end{aligned}$$

Based on the definitions of dependency structure operations and category operations, one can verify the one-to-one correspondence. This correspondence can be formally stated in the following theorem.

Theorem 2
Suppose X and Y are well-formed dependency structures and $\mathrm{OP}(\mathrm{cat}(X), \mathrm{cat}(Y))$ is well-defined. We have

$$\mathrm{cat}(\mathrm{OP}(X, Y)) = \mathrm{OP}(\mathrm{cat}(X), \mathrm{cat}(Y)) \tag{5}$$

Proof
The proof of the theorem is rather routine, so we just give a sketch here. One can show it by induction on the number of nodes in dependency structures. It suffices to show that Equation (5) holds for all the operations. Actually, the category operations are designed to meet this requirement; the three fields of a category represent the head and the children on both sides. ■

With category operations, we can easily track the types of dependency structures and constrain operations in decoding.

Soundness and Completeness

Now we show the soundness and completeness of the operations on dependency structures. If we follow the operations defined herein, we will build all the well-formed structures and only the well-formed structures.

Theorem 3 (Soundness)
Let X and Y be two well-defined dependency structures, and OP an operation over X and Y. It can be shown that $\mathrm{OP}(X, Y)$ is also a well-defined dependency structure.

Proof
Theorem 3 immediately follows Theorem 2. ■

Theorem 4 (Completeness)
Let Z be a well-defined dependency structure with at least two nodes. It can be shown that there exist well-formed structures X, Y and an operation OP, such that $Z = \mathrm{OP}(X, Y)$.

Proof
If Z is fixed on h, without losing generality, we assume g is the leftmost child (or rightmost if there is no left child) of h. We detach g from h, and obtain two sub-trees X and Y which are rooted on g and h respectively. It can be verified that X and Y are well-formed, and $Z = \mathrm{LA}(X, Y)$.

If Z is floating with children $\{c_1, c_2, ..., c_n\}$, where $n > 1$, we can split it into two floating structures with children $\{c_1\}$ and $\{c_2, ..., c_n\}$, respectively. It is easy to verify that they are the sub-structures X and Y that we are looking for. ■

5. Implementation

5.1 Translation Rules

Translation rules are central to an MT system. In our system, each rule translates a source sub-string into a target dependency structure. The target side of the translation rules constitutes a tree grammar.

One way to define a tree grammar is in the way we described earlier. Two well-formed structures can be combined into a larger one with adjoining or concatenation, and there is no non-terminal slot for substitution. This is similar to tree grammars without substitution, such as the original TAG (Joshi, Levy, and Takahashi 1975) and LTAG-spinal (Shen, Champollion, and Joshi 2008). A corresponding MT model was proposed in Carreras and Collins (2009). Search space is a major problem for such an approach, as we described earlier.

In our system, we introduced NT substitution to combat the search problem. The NT slots for substitution come from what we have observed in training data. Combination of well-formed dependency structures can only happen on NT slots. By replacing NT slots with well-formed structures, we implicitly adjoin or concatenate sub-structures based on the dependency information stored in rules. We extract the translation rules from the training data containing word-to-word alignment and target parse trees, which we will explain in the next section. A similar strategy was employed by DeNeefe and Knight (2009). They turned a TAG into an equivalent TIG.

In addition to these extracted rules, we also have special rules to adjoin or concatenate two neighboring hypotheses. Each of the special rules has two NT slots, but they vary on target dependency structures. They are comparable to the glue rules in Chiang (2005).

To formalize translation rules and grammars, a string-to-dependency grammar G is a 4-tuple $G = \langle \mathcal{R}, X, T_f, T_e \rangle$ where $\mathcal{R}$ is a set of transfer rules. X is the only non-terminal *type*.[1] T_f is a set of terminals (words) in the source language, and T_e is a set of terminals in the target language.

A string-to-dependency transfer rule $R \in \mathcal{R}$ is a 4-tuple $R = \langle S_f, S_e, D, A \rangle$ where $S_f \in (T_f \cup \{X\})^+$ is a source string, $S_e \in (T_e \cup \{X\})^+$ is a target string, D represents the dependency structure for S_e, and A is the alignment between S_f and S_e. Non-terminal alignments in A must be one-to-one. We ignore the left hand side for both source and target, since there is only one NT type.

5.2 Rule Extraction

Now we explain how we extract string-to-dependency rules from parallel training data. The procedure is similar to Chiang (2007) except that we maintain tree structures on the target side, instead of strings.

Given sentence-aligned bilingual training data, we first use GIZA++ (Och and Ney 2003) to generate word level alignment. We use a statistical CFG parser to parse

1 Later in the article, we will introduce label information for NTs. However, labels are treated as soft features, and there is still a single NT type. In fact, other useful information can also be treated as soft features, for example, length distribution for each NT observed in the training data. Details are provided in Shen et al. (2009).

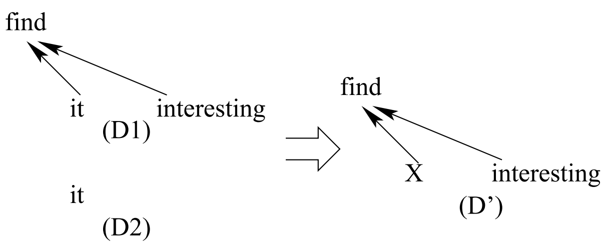

Figure 7
An example to show the rule extraction procedure. In this example, the word *it* is replaced with a non-terminal X, which generates a hierarchical translation rule.

the English side of the training data, and extract dependency trees with Magerman's rules (1995). Then we use heuristic rules to extract transfer rules recursively based on word alignments and the target dependency trees. The rule extraction procedure is as follows.

1. Initialization:
   All the 4-tuples $\langle P_f^{i,j}, P_e^{m,n}, D, A\rangle$ are **valid span templates**, where source phrase $P_f^{i,j}$ is aligned to target phrase $P_e^{m,n}$ under alignment[2] A. D is a well-formed dependency structure for $P_e^{m,n}$. All valid span templates are **valid rule templates**.
2. Inference:
   Let $\langle P_f^{i,j}, P_e^{m,n}, D_1, A\rangle$ be a valid rule template, and $\langle P_f^{p,q}, P_e^{s,t}, D_2, A\rangle$ a valid span template, where range $[p,q] \subset [i,j]$, $[s,t] \subset [m,n]$, D_2 is a sub-structure of D_1, and at least one word in $P_f^{i,j}$ but not in $P_f^{p,q}$ is aligned. We create a new valid rule template $\langle P'_f, P'_e, D', A\rangle$, where we obtain P'_f by replacing $P_f^{p,q}$ with label X in $P_f^{i,j}$, and obtain P'_e by replacing $P_e^{s,t}$ with X in $P_e^{m,n}$. Furthermore, we obtain D' by replacing sub-structure D_2 with X in D_1.[3] An example is shown in Figure 7.

By applying the inference rule recursively, we can generate rules with arbitrary aligned NT slots if there are enough words and alignments. In order to make the size of the grammar manageable, we keep only rules with at most two NT slots and at most seven source elements.

Following previous work (Och and Ney 2003; Chiang 2007), we have three features for each rule, which are P(source|target), P(target|source), and the lexical translation probability given by GIZA. The two conditional probabilities are simply estimated by counting in all the extracted rules.

2 $P_f^{i,j}$ represents the i^{th} to the j^{th} words on the source side, and $P_e^{m,n}$ represents the m^{th} to the n^{th} words on the target side. By $P_f^{i,j}$ aligned to $P_e^{m,n}$, we mean all words in $P_f^{i,j}$ are either aligned to words in $P_e^{m,n}$ or unaligned, and vice versa. Furthermore, at least one word in $P_f^{i,j}$ is aligned to a word in $P_e^{m,n}$.

3 If D_2 is a *floating* structure, we need to merge several dependency links into one.

5.3 Decoding

Following previous work on hierarchical MT (Chiang 2005; Galley et al. 2006), we solve the decoding problem with chart parsing. We view the target dependency trees as hidden structures in the input. The task of decoding is then to find the best hidden structure for the input given the transfer grammar and the language models (a string n-gram LM and a dependency LM).

The parser scans all source cells in a bottom–up style, and checks matched transfer rules according to the source side. Once there is a completed rule, we build a larger dependency structure by substituting component dependency structures for corresponding NTs in the target dependency structure of rules.

Hypotheses, that is, candidate dependency structures, are organized in a shared forest, or AND–OR structures. An AND-structure represents an application of a rule over component OR-structures, and an OR-structure represents a set of alternative AND-structures with the same state. A **state** keeps the necessary information about hypotheses under it, which is needed for computing scores for higher level hypotheses for dynamic programming. For example, with an n-gram string LM in decoding, a state keeps the leftmost $n-1$ words and the rightmost $n-1$ words shared by hypotheses in that state. Because of the use of a dependency LM in decoding, the state information also includes boundary information about dependency structures for the purpose of computing dependency LM scores for larger structures.

In the next section, we will explain how to extend categories and states to exploit a dependency language model during decoding.

5.4 Using Dependency LM Scores

For the dependency tree in Figure 1, we calculate the probability of the tree as follows

$$\begin{aligned} P = & P_T(\text{find}) \\ & \times P_L(\text{will} \mid \text{find-as-head}) \\ & \times P_L(\text{boy} \mid \text{will, find-as-head}) \\ & \times P_L(\text{the} \mid \text{boy-as-head}) \\ & \times P_R(\text{it} \mid \text{find-as-head}) \\ & \times P_R(\text{interesting} \mid \text{it, find-as-head}) \end{aligned}$$

Here $P_T(x)$ is the probability that word x is the root of a dependency tree. P_L and P_R are left and right side generative probabilities respectively. Let w_h be the head, and $w_{L_1}w_{L_2}...w_{L_n}$ be all the children on the left side, from the nearest to the farthest. We use a tri-gram dependency LM,

$$\begin{aligned} P_L(w_{L_1}w_{L_2}...w_{L_n}|w_h\text{-as-head}) = & P_L(w_{L_1}|w_h\text{-as-head}) \times P_L(w_{L_2}|w_{L_1}, w_h\text{-as-head}) \times ... \\ & \times P_L(w_{L_n}|w_{L_{n-1}}, w_{L_{n-2}}) \times P_L(\text{STOP}|w_{L_n}, w_{L_{n-1}}) \end{aligned} \quad (6)$$

In this formula, w_h-as-head represents the event that w is used as the head, and w_{L_i} represents the event that w_{L_i} is a sibling word. The computation of STOP probabilities greatly complicates the implementation of inside dependency LM probabilities, so we ignored it in practice. Right side probability P_R is defined in a similar way.

We should note that other orders of dependency LMs (e.g., bi-gram or 4-gram) can be used by changing the independence assumptions in the above formulas. The choice of using a tri-gram model in our experiments is a trade-off between model robustness and sharpness given the training data available.

In order to calculate the dependency language model score, or depLM score for short, on the fly for partial hypotheses in a bottom–up decoding, we need to save more information in categories and states.

We use a 5-tuple $\langle LF, LN, h, RN, RF \rangle$ to represent the category of a dependency structure. h represents the head. Relative to the head, LF is the farthest children on the left side and RF the farthest children on the right side. Similarly, LN is the nearest children on the left side and RN the nearest children on the right. The three types of categories are as follows.

- fixed: $\langle LF, -, h, -, RF \rangle$
- floating left: $\langle LF, LN, -, -, - \rangle$
- floating right: $\langle -, -, -, RN, RF \rangle$

Furthermore, operations similar to those described in Section 4.2 are used to keep track of the head and boundary child nodes, which are then used to compute depLM scores in decoding.

5.5 Using Labels in Transfer Rules

In the formalism introduced in the previous section, there is only a single non-terminal type X. This may result in loss of information in the training data. For example, there is a rule whose target dependency structure is $X_1 \rightarrow \textit{says} \leftarrow X_2$, where X_1 and X_2 depend on *says*. In the training data, X_1 comes from a tree rooted on a noun and X_2 comes from a tree rooted on a verb. Without this information in the rule, any structure could be placed in either of these two slots in the decoding phase.

We alleviate this problem by associating a label with each non-terminal in the rules. Specifically, each non-terminal has a label, and the whole target structure side also has a label. When we replace an NT with a sub-structure, we check if the label of the sub-structure is the same as the NT label. If they do not match, we assign a penalty to this replacement.

An obvious choice of the label is the POS tag of the head word, if it is a fixed tree. In the previous example, the target structure would generate $X_1(NN) \rightarrow \textit{says} \leftarrow X_2(VBP)$, where *NN* means noun (singular or mass) and *VBP* means verb (non-3rd person singular present), and the whole target structure has a label of *VBZ*, which means verb (3rd person singular present). If we replace *NN* with a sub-tree rooted at, for example, a preposition, there will be a penalty.

In our system, we use the POS tag of the head word as the label of a fixed structure. We always use the generic label for floating structures. Any NT substitution with this label involved is regarded as a mismatch. In other words, there is a penalty for inserting any floating structure during decoding.

This extension does not affect the basic formalism of dependency structures described in the previous section. Instead, we modify the representation of translation rules and states in the decoder. For each rule, if its dependency structure is of a fixed type, the whole structure has a label which is the POS tag of the head word. Otherwise, the label is X. Similarly, each NT slot has a label which is defined in the same way, based on the dependency structure from which the rule is extracted. In decoding, each state has an extra field representing the label for the dependency structure of the hypothesis.

5.6 Other Details

We have nine features in our system.

1. Log probability of the source side given the target side of a rule
2. Log probability of the target side given the source side of a rule
3. Log probability of word alignment
4. Number of target words
5. Number of special rules (see Section 5.1) used
6. Log probability of string LM
7. Log probability of dependency LM
8. Discount on ill-formed dependency structures
9. Discount on unmatched labels

The values of the first four features are accumulated on the rules used in a translation. The fifth feature counts the number of times the adjoining and concatenation rules are used in a translation. The string LM score and dependency LM score are the next two features.

In practice, we also allow hypotheses that do not have well-formed structures in derivation, but they are penalized. For this purpose, we introduce the **null** dependency structure e. For any operation OP and dependency structure X, Y, we have

$$\begin{aligned}\mathrm{OP}(X,Y) &= e \text{ if this operation is not defined in Definition 5}\\ \mathrm{OP}(X,e) &= X\\ \mathrm{OP}(e,X) &= X\end{aligned}$$

Because part of a hypothesis may have a null dependency structure, we cannot calculate dependency LM scores on some of the related words. Therefore, we give a discount for each of these words. This is the eighth feature.

There are two sources for null dependency structures. One is the use of an undefined operation, for example, left-adjoining a right floating structure to a fixed structure. The other source is a lack of target structure information in translation rules. The parser that we used may fail to generate parse trees for short segments—for example, dictionary items. In these cases, we extracted the so-called phrasal rules with null dependency structures. We limited phrasal rules to at most three lexical items for each side.

The last feature counts the number of substitutions with unmatched labels.

In decoding, partial hypotheses are mapped into states. The states maintain sufficient statistics for feature calculation. For example, each state should memorize the leftmost two words and rightmost two words for LM score calculation. Similar extensions are required for dependency LM score and NT labels. Therefore, we use beam search with cube pruning as in Chiang (2005) for speedup. Like chart parsing, the computational complexity of decoding time is $O(n^3 \times B \times |G|)$, where n is the length of the source sentence, B is beam width, and $|G|$ is the maximal number of transfer rules applicable to a span with translation grammar G. This number agrees with the empirical results.

We tune the weights with several rounds of decoding and optimization. Following Och (2003), the k-best results are accumulated as the input to the optimizer. Powell's method is used for optimization with 20 random starting points around the weight vector of the last iteration. For improved results, we rescore 1,000-best translations, generated using the technique described by Huang and Chiang (2005), by replacing tri-gram string LM scores in the output with 5-gram string LM scores. The algorithm to tune the rescoring weights is similar to the one to tune the decoder weights.

6. Experiments

We experimented with four models:

- baseline: hierarchical string to string translation, using our own replication of the Hiero system (Chiang 2007)
- filtered: like the baseline, it uses string to string rules, except that rules whose target side does not correspond to a well-formed structure in rule extraction are excluded. No dependency LM is used in decoding
- str-dep: string-to-dependency system. It uses rules with target dependency structures and a dependency LM in decoding
- labeled: an enhanced str-dep model with POS tags as labels

We use the Hiero model as our baseline because it is the closest to our string-to-dependency model. They use similar rule extraction and decoding algorithms. The major difference is in the representation of target structures. We use dependency structures instead of strings; thus, the comparison will show the contribution of using dependency information in decoding.

All models were tuned on BLEU (Papineni, Roukos, and Ward 2001), and evaluated on BLEU, TER (Snover et al. 2006), and METEOR (Banerjee and Lavie 2005). It is well known that all automatic scores are crude approximations of translation quality. It is not uncommon for a technique to improve the metric that is used for tuning but hurt other metrics. The use of multiple metrics helps us avoid drawing false conclusions based on metric-specific improvements. For both Arabic-to-English and Chinese-to-English MT, we tuned on NIST MT02-05 and tested on MT06 and MT08 newswire sets.

The training data for Arabic-to-English MT contains around 1.9 million pairs of bi-lingual sentences from ten corpora: LDC2004T17, LDC2004T18, LDC2005E46, LDC-2006E25, LDC2006G05, LDC2005E85, LDC2006E36, LDC2006E82, LDC2006E95, and SSUSAC27 (Sakhr Arabic-English Parallel Corpus). The training data for Chinese-to-English MT contains around 1.0 million pairs of bi-lingual sentences from eight corpora: LDC2002E18, LDC2005T06, LDC2005T10, LDC2006E26, LDC2006G05, LDC2002L27, LDC2005T34, and LDC2003E07.

The dependency LMs were trained on the same parallel training data. For that purpose, we parsed the English side of the parallel data. Two separate models were trained: one for Arabic from the Arabic training data and the other for Chinese from the Chinese training data. Traditional tri-gram and 5-gram string LMs were trained on

Table 1
Number of transfer rules.

Model	Arabic-to-English	Chinese-to-English
baseline	337,542,137	193,922,173
filtered	32,057,337	39,005,696
str-dep	35,801,341	41,013,346
labeled	41,201,100	43,705,510

the English side of the parallel data as well as the English Gigaword corpus V3.0 in a way described by Bulyko et al. (2007).

Table 1 shows the number of transfer rules extracted from the training data for the tuning and test sets. The constraint of well-formed dependency structures greatly reduced the size of the rule set. Although the rule size increased a little bit after incorporating dependency structures and labels in rules, the size of string-to-dependency rule set is about 10% to 20% of the baseline.

Tables 2 and 3 show the BLEU, TER, and METEOR scores on MT06 and MT08 for Arabic-to-English MT. Tables 4 and 5 show the scores for Chinese-to-English MT.

For system comparison, we primarily rely on the lower-cased BLEU score of the decoding output because it is the metric on which all systems were tuned. We measured the significance of BLEU, TER, and METEOR with paired bootstrap resampling as proposed by Koehn (2004). In Tables 2 through 5, (+/−) represent being better/worse than the baseline at 95% confidence level, respectively, and (∗) represents insignificant difference from the baseline.

For Arabic-to-English MT, the str-dep model decoder improved BLEU by 1.3 on MT06 and 1.2 on MT08 before 5-gram rescoring. For Chinese-to-English MT, the improvements in BLEU were 1.0 on MT06 and 1.4 on MT08. After rescoring, the improvements became smaller, ranging from 0.8 to 1.3. All the BLEU improvements on 5-gram scores are statistically significant.

The use of POS labels in transfer rules further improves the BLEU score by about 0.7 points on average. The overall BLEU improvement on lower-cased decoding output

Table 2
BLEU, TER, and METEOR percentage scores on MT06 Arabic-to-English newswire set.

Model	BLEU		TER		METEOR
	lower	mixed	lower	mixed	
		Decoding (3-gram LM)			
baseline	47.50	45.48	44.79	46.97	66.17
filtered	46.64 (−)	44.47 (−)	45.38 (∗)	47.96 (−)	66.64 (∗)
str-dep	48.75 (+)	46.74 (+)	43.43 (+)	45.79 (+)	67.18 (+)
labeled	49.33 (+)	47.07 (+)	43.09 (+)	45.53 (+)	67.04 (+)
		Rescoring (5-gram LM)			
baseline	50.38	48.33	42.64	44.87	67.25
filtered	49.60 (−)	47.51 (−)	43.50 (−)	45.81 (−)	67.44 (∗)
str-dep	51.24 (+)	49.23 (+)	42.08 (∗)	44.42 (∗)	67.89 (+)
labeled	51.80 (+)	49.69 (+)	41.54 (+)	43.76 (+)	67.97 (+)

Table 3
BLEU, TER, and METEOR percentage scores on MT08 Arabic-to-English newswire set.

Model	BLEU		TER		METEOR
	lower	mixed	lower	mixed	
			Decoding (3-gram LM)		
baseline	48.41	46.13	43.83	46.18	67.45
filtered	47.37 (−)	45.24 (−)	44.39 (−)	46.83 (−)	67.17 (∗)
str-dep	49.58 (+)	47.46 (+)	42.80 (+)	45.08 (+)	68.08 (+)
labeled	50.46 (+)	48.19 (+)	42.27 (+)	44.57 (+)	67.78 (+)
			Rescoring (5-gram LM)		
baseline	50.50	48.35	42.78	44.92	67.98
filtered	49.56 (−)	47.49 (−)	43.20 (∗)	45.44 (∗)	67.79 (∗)
str-dep	51.23 (+)	49.11 (+)	42.01 (+)	44.15 (+)	68.65 (+)
labeled	51.93 (+)	49.86 (+)	41.27 (+)	43.33 (+)	68.40 (+)

is 1.8 points on MT06 and 2.1 points on MT08 for Arabic-to-English translation, and 2.0 points on MT06 and 1.6 points on MT08 for Chinese-to-English translation.

METEOR scores became significantly better for all conditions. TER improved significantly for Arabic-to-English but marginally on Chinese-to-English tasks. The results on METEOR and TER suggested that the new model did improve translation accuracy.

The filtered string-to-string rules can be viewed as the string projection of string-to-dependency rules. It shows the performance of using dependency structure for rule filtering only. The results are very interesting. On Arabic-to-English, the filtered model was significantly worse, which means that many useful rules were lost due to the structural constraints. On Chinese-to-English, the tri-gram scores of the filtered model were a little bit worse. However, after 5-gram rescoring, the BLEU scores became higher than the baseline, and METEOR scores were even significantly better. We suspect that the different performance that we observed is due to the difference in source languages and their tokenization methods.

Table 4
BLEU, TER, and METEOR percentage scores on MT06 Chinese-to-English newswire set.

Model	BLEU		TER		METEOR
	lower	mixed	lower	mixed	
			Decoding (3-gram LM)		
baseline	36.40	34.79	54.98	56.53	57.25
filtered	36.02 (∗)	34.23 (∗)	55.29 (∗)	57.03 (∗)	57.60 (+)
str-dep	37.44 (+)	35.62 (+)	54.64 (∗)	56.47 (∗)	57.42 (+)
labeled	38.37 (+)	36.53 (+)	54.14 (+)	55.99 (∗)	58.42 (+)
			Rescoring (5-gram LM)		
baseline	37.88	36.18	53.80	55.45	57.44
filtered	38.52 (∗)	36.74 (∗)	54.09 (∗)	55.69 (∗)	58.16 (+)
str-dep	38.91 (+)	37.04 (+)	53.65 (∗)	55.45 (∗)	57.99 (+)
labeled	39.11 (+)	37.30 (+)	53.61 (∗)	55.29 (∗)	58.69 (+)

Table 5
BLEU, TER, and METEOR percentage scores on MT08 Chinese-to-English newswire set.

Model	BLEU		TER		METEOR
	lower	mixed	lower	mixed	
		Decoding (3-gram LM)			
baseline	31.64	29.56	57.35	59.37	54.93
filtered	31.26 (∗)	29.42 (∗)	57.46 (∗)	59.28 (∗)	55.16(+)
str-dep	33.05 (+)	31.26 (+)	56.79 (∗)	58.69 (+)	55.18(+)
labeled	33.25 (+)	31.34 (+)	56.60 (+)	58.49 (+)	56.01(+)
		Rescoring (5-gram LM)			
baseline	33.06	31.21	55.84	57.71	55.18
filtered	33.25 (∗)	31.22 (∗)	56.53 (∗)	58.39 (−)	56.08 (+)
str-dep	34.34 (+)	32.32 (+)	55.60 (∗)	57.60 (∗)	55.91 (+)
labeled	35.02 (+)	33.00 (+)	55.39 (∗)	57.48 (∗)	56.46 (+)

In any case, the purpose of the filtered model is not to propose the use of structural constraints for rule filtering, although it greatly reduced the rule size and allowed the use of more useful training data potentially. The use of structural constraints is compulsory for the introduction of dependency LMs and non-terminal labels, which compensated for the loss of rule filtering, and led to significant overall improvement.

7. Comparison to Related Work

Fox (2002), Ding and Palmer (2005), and Quirk, Menezes, and Cherry (2005) showed that, for the purpose of representing word relations, dependency structures are advantageous over CFG structures because they do not require complete constituents. A number of techniques have been proposed to improve rule coverage. Marcu et al. (2006) and Galley et al. (2006) introduced artificial constituent nodes dominating the phrase of interest. The binarization method used by Wang, Knight, and Marcu (2007) can cover many non-constituent rules also, but not all of them. DeNeefe et al. (2007) showed that the best results were obtained by combining these methods.

Charniak, Knight, and Yamada (2003) described a two-step string-to-CFG-tree translation model which employed a syntax-based language model to select the best translation from a target parse forest built in the first step. A crucial difference from our work is that they only used the tree-based LM in rescoring, possibly due to the complexity of the syntax-based LM. In contrast, our system uses a dependency LM directly in decoding and as such can prune out unpromising hypotheses as soon as possible.

The use of a dependency LM in MT is similar to the use of a structured LM in ASR (Chelba and Jelinek 2000; Xu, Chelba, and Jelinek 2002), with the same motivation of exploiting long-distance relations. A difference is that the dependency LM is used bottom–up in our MT system, whereas the structured LM is used left-to-right in ASR. Another difference is that long-distance relations are more important in MT due to word re-orderings.

The well-formed dependency structures defined here are similar to the data structures in previous work on monolingual parsing (Eisner and Satta 1999; McDonald, Crammer, and Pereira 2005), which allowed floating structures as well defined states in derivation, too. However, as for monolingual parsing, one usually wants exactly one derivation for each parse tree, so as to avoid spurious ambiguity of derivations for the same parse. The derivation model proposed by Eisner and Satta (1999) satisfied this prerequisite, and had $O(n^3)$ complexity with a bi-lexical probability model, which was $O(n^4)$ in many other derivation models. In our MT model, the motivation is to exploit various translation fragments learned from the training data, and the operations in monolingual parsing were designed to avoid artificial ambiguity of derivation. Another difference is that we have fixed structures growing on both sides, whereas fixed structures in (Eisner and Satta 1999) can only grow in one direction.

The formalism for well-formed structures and the operations over them were inspired by the well-known approach of Combinatory Categorial Grammar (CCG) (Steedman 2000). In fact, the names of left raising and right raising stem from the raising operation in CCG.

The string-to-dependency formalism can be viewed as a special case of Synchronous Tree Adjoining Grammar (STAG) (Shieber and Schabes 1990). Trees on the source side are weakened to strings, and multi-rooted structures are employed on the target side. The adjoining operation in our model is similar to attachment in LTAG-spinal (Shen, Champollion, and Joshi 2008) and sister adjunction in variants (Rambow, Shanker, and Weir 1995; Chiang 2000; Carreras, Collins, and Koo 2008) of TAG (Joshi and Schabes 1997). Translation rules can be viewed as constraints on the tree operations.

8. Conclusions and Future Work

In this article, we propose a novel string-to-dependency algorithm for statistical machine translation. It employs a target dependency language model to exploit long distance word relations in decoding, which cannot be captured with a traditional *n*-gram language model.

Compared with a state-of-the-art hierarchical string-to-string system, our string-to-dependency system generates about 80% fewer rules. The overall gain in BLEU score on lower-cased decoding output is about two points.

Dependency structures provide a desirable platform for employing linguistic knowledge in MT. We will extend our approach with deeper linguistic features such as propositional structures (Palmer, Gildea, and Kingsbury 2005). The fixed and floating structures proposed in this article can be extended to model predicates and arguments.

Acknowledgments
This work[4] was supported by DARPA/IPTO Contract No. HR0011-06-C-0022 under the GALE program. The views, opinions, and/or findings contained in this article/presentation are those of the author/presenter and should not be interpreted as representing the official views or policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the Department of Defense.
We are grateful to our colleagues Roger Bock, Ivan Bulyko, Mike Kayser, Jeff Ma, John Makhoul, Spyros Matsoukas, Antti-Veikko Rosti, Rich Schwartz, Bing Zhang, and Rabih Zbib for their help in running the experiments and constructive comments to improve this article. Mike Kayser helped to proofread the manuscript. We also thank Zhifei Li and the anonymous reviewers for their suggestions to improve this article.

4 Distribution Statement "A" (Approved for Public Release, Distribution Unlimited).

References

Banerjee, Satanjeev and Alon Lavie. 2005. Meteor: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 101–104, Ann Arbor, MI.

Bulyko, Ivan, Spyros Matsoukas, Richard Schwartz, Long Nguyen, and John Makhoul. 2007. Language model adaptation in machine translation from speech. In *Proceedings of the 32nd IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP)*, pages 117–120, Honolulu, HI.

Carreras, Xavier and Michael Collins. 2009. Non-projective parsing for statistical machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 200–209, Singapore.

Carreras, Xavier, Michael Collins, and Terry Koo. 2008. TAG, dynamic programming, and the perceptron for efficient, feature-rich parsing. In *Proceedings of the 12th Conference on Computational Natural Language Learning*, pages 9–16, Manchester.

Charniak, Eugene, Kevin Knight, and Knight Yamada. 2003. Syntax-based language models for statistical machine translation. In *Proceedings of MT Summit IX*, pages 40–46, New Orleans, LA.

Chelba, Ciprian and Frederick Jelinek. 2000. Structured language modeling. *Computer Speech and Language*, 14(4):283–332.

Chiang, David. 2000. Statistical parsing with an automatically extracted tree adjoining grammar. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 456–463, Hong Kong.

Chiang, David. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 263–270, Ann Arbor, MI.

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

DeNeefe, Steve and Kevin Knight. 2009. Synchronous tree adjoining machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 727–736, Singapore.

DeNeefe, Steve, Kevin Knight, Wei Wang, and Daniel Marcu. 2007. What can syntax-based MT learn from phrase-based MT? In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*, pages 755–763, Prague.

Ding, Yuan and Martha Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 541–548, Ann Arbor, MI.

Eisner, Jason. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 205–208, Sapporo.

Eisner, Jason and Giorgio Satta. 1999. Efficient parsing for bilexical context-free grammars and head automaton grammars. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 457–464, College Park, MD.

Fox, Heidi. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of the 2002 Conference of Empirical Methods in Natural Language Processing*, pages 304–311, Philadelphia, PA.

Galley, Michel, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic models. In *COLING-ACL '06: Proceedings of 44th Annual Meeting of the Association for Computational Linguistics and 21st International Conference on Computational Linguistics*, pages 961–968, Sydney.

Galley, Michel, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 273–280, Boston, MA.

Graehl, Jonathan and Kevin Knight. 2004. Training tree transducers. In *Proceedings of the 2004 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 105–112, Boston, MA.

Hajič, Jan, Martin Čmejrek, Jason Eisner, Gerald Penn, Owen Rambow, Dragomir Radev, Yuan Ding, Terry Koo, and Kristen Parton. 2002. Natural language generation in the context of machine translation. Final report of JHU Summer Workshop project, Johns Hopkins University, Baltimore, MD.

Huang, Liang and David Chiang. 2005. Better k-best parsing. In *Proceedings of the 9th International Workshop on Parsing Technologies*, pages 53–64, Vancouver.

Joshi, Aravind K., Leon S. Levy, and Masako Takahashi. 1975. Tree adjunct grammars.

Journal of Computer and System Sciences, 10(1):136–163.

Joshi, Aravind K. and Yves Schabes. 1997. Tree-adjoining grammars. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Languages*, volume 3. Springer-Verlag, Berlin, pages 69–124.

Koehn, Philipp. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference of Empirical Methods in Natural Language Processing*, pages 388–395, Barcelona.

Koehn, Philipp, Franz J. Och, and Daniel Marcu. 2003. Statistical phrase based translation. In *Proceedings of the 2003 Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–54, Edmonton.

Magerman, David. 1995. Statistical decision-tree models for parsing. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 276–283, Cambridge, MA.

Marcu, Daniel, Wei Wang, Abdessamad Echihabi, and Kevin Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proceedings of the 2006 Conference of Empirical Methods in Natural Language Processing*, pages 44–52, Sydney.

McDonald, Ryan, Koby Crammer, and Fernando Pereira. 2005. Online large-margin training of dependency parsers. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 91–98, Ann Arbor, MI.

Mi, Haitao, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 192–199, Columbus, OH.

Och, Franz J. 2003. Minimum error rate training for statistical machine translation. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 160–167, Sapporo.

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

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–106.

Papineni, Kishore, Salim Roukos, and Todd Ward. 2001. BLEU: A method for automatic evaluation of machine translation. IBM Research Report No. RC22176, Armonk, NY.

Quirk, Chris, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal SMT. In *Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 271–279, Ann Arbor, MI.

Rambow, Owen, Vijay K. Shanker, and David Weir. 1995. D-tree grammars. In *Proceedings of the 33rd Annual Meeting of the Association for Computational Linguistics*, pages 151–158, Cambridge, MA.

Shen, Libin, Lucas Champollion, and Aravind K. Joshi. 2008. LTAG-spinal and the Treebank: A new resource for incremental, dependency and semantic parsing. *Language Resources and Evaluation*, 42(1):1–19.

Shen, Libin and Aravind K. Joshi. 2005. Incremental LTAG Parsing. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 811–818, Vancouver.

Shen, Libin and Aravind K. Joshi. 2008. LTAG dependency parsing with bidirectional incremental construction. In *Proceedings of the 2008 Conference of Empirical Methods in Natural Language Processing*, pages 495–504, Honolulu, HI.

Shen, Libin, Jinxi Xu, Bing Zhang, Spyros Matsoukas, and Ralph Weischedel. 2009. Effective use of linguistic and contextual information for statistical machine translation. In *Proceedings of the 2009 Conference of Empirical Methods in Natural Language Processing*, pages 72–80, Singapore.

Shieber, Stuart and Yves Schabes. 1990. Synchronous tree adjoining grammars. In *Proceedings of COLING '90: The 13th International Conference on Computational Linguistics*, pages 253–258, Helsinki.

Smith, David A. and Jason Eisner. 2006. Quasi-synchronous grammars: Alignment by soft projection of syntactic dependencies. In *Proceedings of the HLT-NAACL Workshop on Statistical Machine Translation*, pages 23–30, New York, NY.

Snover, Matthew, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of Association for Machine Translation in the Americas*, pages 223–231, Cambridge, MA.

Steedman, Mark. 2000. *The Syntactic Process*. The MIT Press, Cambridge, MA.

Wang, Wei, Kevin Knight, and Daniel Marcu. 2007. Binarizing syntax trees to improve syntax-based machine translation accuracy. In *Proceedings of the 2007 Conference of Empirical Methods in Natural Language Processing*, pages 746–754, Prague.

Xu, Peng, Ciprian Chelba, and Frederick Jelinek. 2002. A study on richer syntactic dependencies for structured language modeling. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 191–198, Philadelphia, PA.

Yamada, Kenji and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 523–530, Toulouse.