Transformation and Decomposition for Efficiently Implementing and Improving Dependency-to-String Model In Moses

Liangyou Li[†], Jun Xie[‡], Andy Way[†] and Qun Liu[†‡]
[†] CNGL Centre for Global Intelligent Content, School of Computing
Dublin City University, Dublin 9, Ireland
[‡] Key Laboratory of Intelligent Information Processing, Institute of Computing Technology
Chinese Academy of Sciences, Beijing, China
{liangyouli,away,qliu}@computing.dcu.ie
junxie@ict.ac.cn

Abstract

Dependency structure provides grammatical relations between words, which have shown to be effective in Statistical Machine Translation (SMT). In this paper, we present an open source module in Moses which implements a dependency-to-string model. We propose a method to transform the input dependency tree into a corresponding constituent tree for reusing the tree-based decoder in Moses. In our experiments, this method achieves comparable results with the standard model. Furthermore, we enrich this model via the decomposition of dependency structure, including extracting rules from the substructures of the dependency tree during training and creating a pseudo-forest instead of the tree per se as the input during decoding. Large-scale experiments on Chinese–English and German–English tasks show that the decomposition approach improves the baseline dependency-to-string model significantly. Our system achieves comparable results with the state-of-the-art hierarchical phrase-based model (HPB). Finally, when resorting to phrasal rules, the dependency-to-string model performs significantly better than Moses HPB.

1 Introduction

Dependency structure models relations between words in a sentence. Such relations indicate the syntactic function of one word to another word. As dependency structure directly encodes semantic information and has the best inter-lingual phrasal cohesion properties (Fox, 2002), it is believed to be helpful to translation.

In recent years, dependency structure has been widely used in SMT. For example, Shen et al. (2010) present a string-to-dependency model by using the dependency fragments of the neighbouring words on the target side, which makes it easier to integrate a dependency language model. However such string-to-tree systems run slowly in cubic time (Huang et al., 2006).

Another example is the treelet approach (Menezes and Quirk, 2005; Quirk et al., 2005), which uses dependency structure on the source side. Xiong et al. (2007) extend the treelet approach to allow dependency fragments with gaps. As the treelet is defined as an arbitrary connected sub-graph, typically both substitution and insertion operations are adopted for decoding. However, as translation rules based on the treelets do not encode enough reordering information directly, another heuristic or separate reordering model is usually needed to decide the best target position of the inserted words.

Different from these works, Xie et al. (2011) present a dependency-to-string (Dep2Str) model, which extracts head-dependent (HD) rules from word-aligned source dependency trees and target strings. As this model specifies reordering information in the HD rules, during translation only the substitution operation is needed, because words are reordered simultaneously with the rule being applied. Meng et al. (2013) and Xie et al. (2014) extend the model by augmenting HD rules with the help of either constituent tree or fixed/float structure (Shen et al., 2010). Augmented rules are created by the combination of two or more nodes in the HD fragment, and are capable of capturing translations of non-syntactic phrases. However, the decoder needs to be changed correspondingly to handle these rules.

Attracted by the simplicity of the Dep2Str model, in this paper we describe an easy way to integrate the model into the popular translation framework Moses (Koehn et al., 2007). In order to share the same decoder with the conventional syntax-based model, we present an algorithm which transforms a dependency tree into a corresponding constituent tree which encodes dependency information in its non-leaf nodes and is compatible with the Dep2Str model. In addition, we present a method to decompose a dependency structure (HD fragment) into smaller parts which enrich translation rules and also allow us to create a pseudo-forest as the input. “Pseudo” means the forest is not obtained by combining several trees from a parser, but rather that it is created based on the decomposition of an HD fragment. Large-scale experiments on Chinese–English and German–English tasks show that the transformation and decomposition are effective for translation.

In the remainder of the paper, we first describe the Dep2Str model (Section 2). Then we describe how to transform a dependency tree into a constituent tree which is compatible with the Dep2Str model (Section 3). The idea of decomposition including extracting sub-structural rules and creating a pseudo-forest is presented in Section 4. Then experiments are conducted to compare translation results of our approach with the state-of-the-art HPB model (Section 5). We conclude in Section 6 and present avenues for future work.

2 Dependency-to-String Model

In the Dep2Str model (Xie et al., 2011), the HD fragment is the basic unit. As shown in Figure 1, in a dependency tree, each non-leaf node is the head of some other nodes (dependents), so an HD fragment is composed of a head node and all of its dependents.[1]

In this model, there are two kinds of rules for translation. One is the head rule which specifies the translation of a source word:

$$\overset{\text{Juxing}}{\text{举行}} \rightarrow \text{holds}$$

[1]In this paper, HD fragment of a node means the HD fragment with this node as the head. Leaf nodes have no HD fragments.

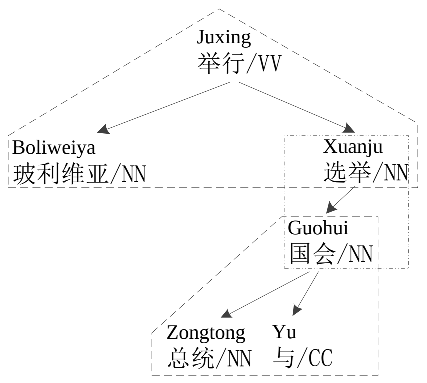


Figure 1: Example of a dependency tree, with head-dependent fragments being indicated by dotted lines.

The other one is the HD rule which consists of three parts: the HD fragment s of the source side (maybe containing variables), a target string t (maybe containing variables) and a one-to-one mapping ϕ from variables in s to variables in t, as in:

$$s = (\overset{\text{Boliweiya}}{\underline{\text{玻利维亚}}})\ \overset{\text{Juxing}}{\text{举行}}\ (x_1\text{:}\ \overset{\text{Xuanju}}{\text{选举}}\ )$$

$$t = \text{Bolivia holds } x_1$$

$$\phi = \{x_1\ \text{:}\overset{\text{Xuanju}}{\text{选举}} \rightarrow x_1\}$$

where the underlined element denotes the leaf node. Variables in the Dep2Str model are constrained either by words (like x_1:选举) or Part-of-Speech (POS) tags (like x_1:NN).

Given a source sentence with a dependency tree, a target string and the word alignment between the source and target sentences, this model first annotates each node N with two annotations: head span and dependency span.[2] These two spans specify the corresponding target position of a node (by the head span) or sub-tree (by the dependency span). After annotation, acceptable HD fragments[3] are utilized to induce lexicalized HD

[2]Some definitions: Closure $clos(S)$ of set S is the smallest superset of S in which the elements (integers) are continuous. Let H be the set of indexes of target words aligned to node N. Head span $hsp(N)$ of node N is $clos(H)$. Head span $hsp(N)$ is $consistent$ if it does not overlap with head span of any other node. Dependency span $dsp(N)$ of node N is the union of all $consistent$ head spans in the subtree rooted at N.

[3]A head-dependent fragment is acceptable if the head span of the head node is $consistent$ and none of the dependency spans of its dependents is empty. We could see that in an acceptable fragment, the head span of the head node and dependency spans of dependents are not overlapped with each other.

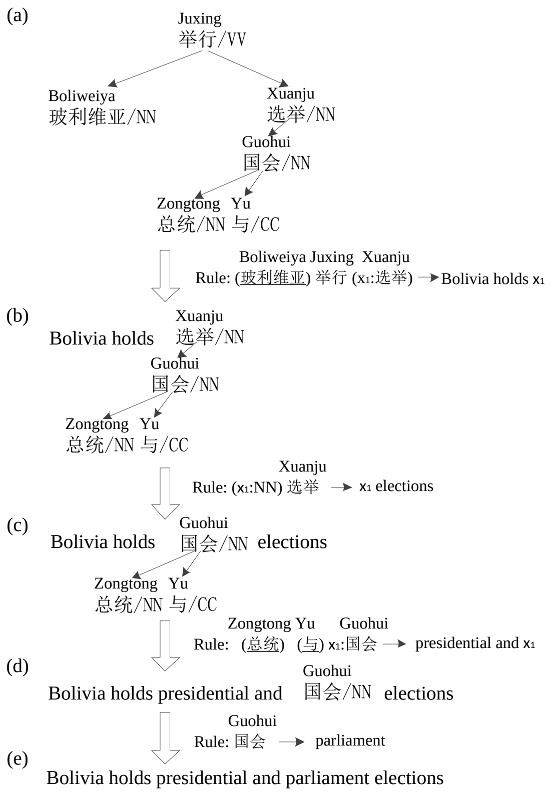


Figure 2: Example of a derivation. Underlined elements indicate leaf nodes.

rules (the head node and leaf node are represented by words, while the internal nodes are replaced by variables constrained by word) and unlexicalized HD rules (nodes are replaced by variables constrained by POS tags).

In HD rules, an internal node denotes the whole sub-tree and is always a substitution site. The head node and leaf nodes can be represented by either words or variables. The target side corresponding to an HD fragment and the mapping between variables are determined by the head span of the head node and the dependency spans of the dependents.

A translation can be obtained by applying rules to the input dependency tree. Figure 2 shows a derivation for translating a Chinese sentence into an English string. The derivation proceeds from top to bottom. Variables in the higher-level HD rules are substituted by the translations of lower HD rules recursively.

The final translation is obtained by finding the best derivation d^* from all possible derivations D which convert the source dependency structure into a target string, as in Equation (1):

$$d^* = \operatorname*{argmax}_{d \in D} p(d) \approx \operatorname*{argmax}_{d \in D} \prod_i \phi_i(d)^{\lambda_i} \quad (1)$$

where $\phi_i(d)$ is the ith feature defined in the derivation d, and λ_i is the weight of the feature.

3 Transformation of Dependency Trees

In this section, we introduce an algorithm to transform a dependency tree into a corresponding constituent tree, where words of the source sentence are leaf nodes and internal nodes are labelled with head words or POS tags which are constrained by dependency information. Such a transformation makes it possible to use the traditional tree-based decoder to translate a dependency tree, so we can easily integrate the Dep2Str model into the popular framework Moses.

In a tree-based system, the CYK algorithm (Kasami, 1965; Younger, 1967; Cocke and Schwartz, 1970) is usually employed to translate the input sentence with a tree structure. Each time a continuous sequence of words (a phrase) in the source sentence is translated. Larger phrases can be translated by combining translations of smaller phrases.

In a constituent tree, the source words are leaf nodes and all non-leaf nodes covering a phrase are labelled with categories which are usually variables defined in the tree-based model. For translating a phrase covered by a non-leaf node, the decoder for the constituent tree can easily find applied rules by directly matching variables in these rules to tree nodes. However, in a dependency tree, each internal node represents a word of the source sentence. Variables covering a phrase cannot be recognized directly. Therefore, to share the same decoder with the constituent tree, the dependency tree needs to be transformed into a constituent-style tree.

As we described in Section 2, each variable in the Dep2Str model represents a word (for the head and leaf node) or a sequence of continuous words (for the internal node). Thus it is intuitive to use these variables to label non-leaf nodes of the produced constituent tree. Furthermore, in order to preserve the dependency information of each HD fragment, the created constituent node needs to be constrained by the dependency information in the HD fragment.

Our transformation algorithm is shown in Algorithm 1, which proceeds recursively from top to bottom on each HD fragment. There are a maximum of three types of nodes in an HD fragment: head node, leaf nodes, and internal nodes. The

Algorithm 1 Algorithm for transforming a dependency tree to constituent tree. Dnode means node in dependency tree. Cnode means node in constituent tree.

```
function CNODE($label$, $span$)
    create a new Cnode $CN$
    $CN.label \leftarrow label$
    $CN.span \leftarrow span$
end function
function TRANSFNODE(Dnode $H$)
    $pos \leftarrow$ POS of $H$
    constrain $pos$                    ▷ with H0, like: NN:H0
    CNODE($label$,$H.position$)
    for each dependent $N$ of $H$ do
        $pos \leftarrow$ POS of $N$
        $word \leftarrow$ word of $N$
        constrain $pos$          ▷ with L$i$ or R$i$, like: NN:R1
        constrain $word$                    ▷ with L$i$ or R$i$
        if $N$ is leaf then
            CNODE($pos$,$N.position$)
        else
            CNODE($word$,$H.span$)
            CNODE($pos$,$H.span$)
            TRANSFNODE($N$)
        end if
    end for
end function
```

leaf nodes and internal nodes are dependents of the head node. For the leaf node and head node, we create constituent nodes that just cover one word. For an internal node N, we create constituent nodes that cover all the words in the sub-tree rooted at N. In Algorithm 1, $N.position$ means the position of the word represented by the node N. $N.span$ denotes indexes of words covered by the sub-tree rooted at node N.

Taking the dependency tree in Figure 1 as an example, its transformation result for integration with Moses is shown in Figure 3. In the Dep2Str model, leaf nodes can be replaced by a variable constrained by its POS tag, so for leaf node “$\overset{\text{Zongtong}}{\text{总统}}$” in HD fragment “($\overset{\text{Zongtong}}{\underline{\text{总统}}}$) ($\overset{\text{Yu}}{\underline{\text{与}}}$) $\overset{\text{Guohui}}{\text{国会}}$”, we create a constituent node “NN:L2”, where “NN” is the POS tag and “L2” denotes that the leaf node is the second left dependent of the head node. For the internal node “$\overset{\text{Guohui}}{\text{国会}}$” in the HD fragment “($\overset{\text{Guohui}}{\text{国会}}$) $\overset{\text{Xuanju}}{\text{选举}}$”, we create two constituent nodes

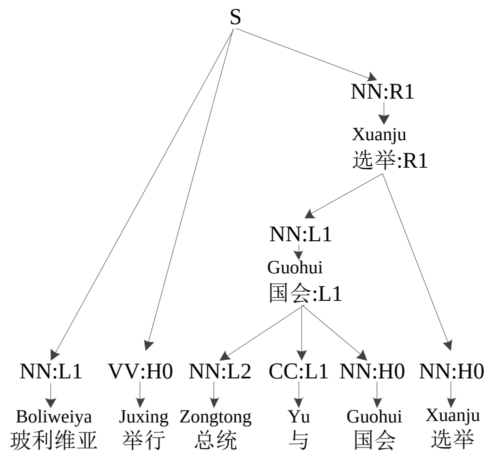


Figure 3: The corresponding constituent tree after transforming the dependency tree in Figure 1. Note in our implementation, we do not distinguish the leaf node and internal node of a dependency tree in the produced constituent tree and induced rules.

which cover all words in the dependency sub-tree rooted at this node, with one of them labelled by the word itself. Both nodes are constrained by dependency information “L1”. After such a transformation is conducted on each HD fragment recursively, we obtain a constituent tree.

This transformation makes our implementation of the Dep2Str model easier, because we can use the tree-to-string decoder in Moses. All we need to do is to write a new rule extractor which extracts head rules and HD rules (see Section 2) from the word-aligned source dependency trees and target strings, and represents these rules in the format defined in Moses.[4]

Note that while this conversion is performed on an input dependency tree during decoding, the training part, including extracting rules and calculating translation probabilities, does not change, so the model is still a dependency-to-string model.

[4]Taking the rule in Section 2 as an example, its representation in Moses is:

$$s = \overset{\text{Boliweiya}}{\text{玻利维亚}}\ \overset{\text{Juxing}}{\text{举行}}\ [\overset{\text{Xuanju}}{\text{选举}}\text{:R1}][\text{X}]\ [\text{H1}]$$
$$t = \text{Bolivia holds } [\overset{\text{Xuanju}}{\text{选举}}\text{:R1}][\text{X}]\ [\text{X}]$$
$$\phi = \{2 \rightarrow 2\}$$

where “H1” denotes the position of the head word is 1, “R1” indicates the first right dependent of the head word, “X” is the general label for the target side and ϕ is the set of alignments (the index-correspondences between s and t). The format has been described in detail at `http://www.statmt.org/moses/?n=Moses.SyntaxTutorial`.

In addition, our transformation is different from other works which transform a dependency tree into a constituent tree (Collins et al., 1999; Xia and Palmer, 2001). In this paper, the produced constituent tree still preserves dependency relations between words, and the phrasal structure is directly derived from the dependency structure without refinement. Accordingly, the constituent tree may not be a linguistically well-formed syntactic structure. However, it is not a problem for our model, because in this paper what matters is the dependency structure which has already been encoded into the (ill-formed) constituent tree.

4 Decomposition of Dependency Structure

The Dep2Str model treats a whole HD fragment as the basic unit, which may result in a sparse-data problem. For example, an HD fragment with a verb as head typically consists of more than four nodes (Xie et al., 2011). Thus in this section, inspired by the treelet approach, we describe a decomposition method to make use of smaller fragments.

In an HD fragment of a dependency tree, the head determines the semantic category, while the dependent gives the semantic specification (Zwicky, 1985; Hudson, 1990). Accordingly, it is reasonable to assume that in an HD fragment, dependents could be removed or new dependents could be attached as needed. Thus, in this paper, we assume that a large HD fragment is formed by attaching dependents to a small HD fragment. For simplicity and reuse of the decoder, such an attachment is carried out in one step. This means that an HD fragment is decomposed into two smaller parts in a possible decomposition. This decomposition can be formulated as Equation (2):

$$\begin{aligned}
&L_i \cdots L_1 H R_1 \cdots R_j \\
&\qquad = L_m \cdots L_1 H R_1 \cdots R_n \\
&\qquad + L_i \cdots L_{m+1} H R_{n+1} \cdots R_j \\
&\text{subject to} \qquad\qquad\qquad\qquad (2) \\
&\qquad i \geq 0, j \geq 0 \\
&\qquad i \geq m \geq 0, j \geq n \geq 0 \\
&\qquad i + j > m + n > 0
\end{aligned}$$

where H denotes the head node, L_i denotes the ith left dependent and R_j denotes the jth right dependent. Figure 4 shows an example.

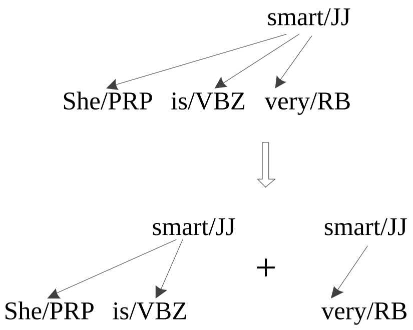


Figure 4: An example of decomposition on a head-dependent fragment.

Algorithm 2 Algorithm for the decomposition of an HD fragment into two sub-fragments. Index of nodes in a fragment starts from 0.

```
function DECOMP(HD fragment $frag$)
    $fset \leftarrow \{\}$
    $len \leftarrow$ number of nodes in $frag$
    $hidx \leftarrow$ the index of head node in $frag$
    for $s$ = 0 to $hidx$ do
        for $e$ = $hidx$ to $len - 1$ do
            if $0 < e - s < len - 1$ then
                create sub-fragment $core$
                $core \leftarrow$ nodes from $s$ to $e$
                add $core$ to $fset$
                create sub-fragment $shell$
                initialize $shell$ with head node
                $shell \leftarrow$ nodes not in $core$
                add $shell$ to $fset$
            end if
        end for
    end for
end function
```

Such a decomposition of an HD fragment enables us to create translation rules extracted from sub-structures and create a pseudo-forest from the input dependency tree to make better use of smaller rules.

4.1 Sub-structural Rules

In the Dep2Str model, rules are extracted on an entire HD fragment. In this paper, when the decomposition is considered, we also extract sub-structural rules by taking each possible sub-fragment as a new HD fragment. The algorithm for recognizing the sub-fragments is shown in Algorithm 2.

In Algorithm 2, we find all possible decompositions of an HD fragment. Each decomposition produces two sub-fragments: *core* and *shell*. Both *core* and *shell* include the head node. *core* contains the dependents surrounding the head node, with the remaining dependents belonging to *shell*. Taking Figure 4 as an example, the bottom-right part is *core*, while the bottom-left part is *shell*. Each *core* and *shell* could be seen as a new HD fragment. Then HD rules are extracted as defined in the Dep2Str model.

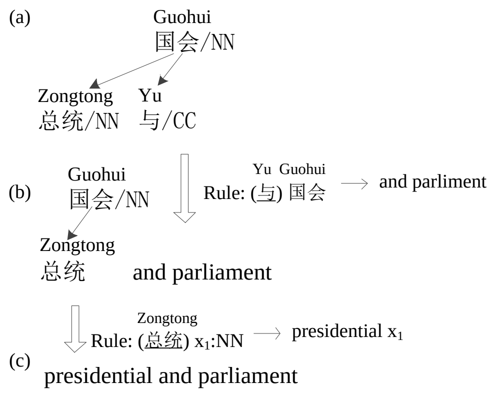
Note that different from the augmented HD rules, where Meng et al. (2013) annotate rules with combined variables and Xie et al. (2014) create special rules from HD rules at runtime by combining several nodes, our sub-structural rules are standard HD rules, which are extracted from the connected sub-structures of a larger HD fragment and can be used directly in the model.

4.2 Pseudo-Forest

Although sub-structural rules are effective in our experiments (see Section 5), we still do not use them to their best advantage, because we only enrich smaller rules in our model. During decoding, for a large input HD fragment, the model is still more likely to resort to glue rules. However, the idea of decomposition allows us to create a pseudo-forest directly from the dependency tree to alleviate this problem to some extent.

As described above, an HD fragment can be seen as being created by combining two smaller fragments. This means, for an HD fragment in the input dependency tree, we can translate one of its sub-fragments first, then obtain the whole translation by combining with translations of another sub-fragment. From Algorithm 2, we know that the sub-fragment *core* covers a continuous phrase of the source sentence. Accordingly, we can translate this fragment first and then build the whole translation by translating another sub-fragment *shell*. Figure 5 gives an example of translating an HD fragment by combining the translations of its sub-fragments.

Instead of taking the dependency tree as the input and looking for all rules for translating sub-fragments of a whole HD, we directly encode the decomposition into the input dependency tree with the result being a pseudo-forest. Based on the transformation algorithm in Section 3, the pseudo-forest can also be represented in the constituent-tree style, as shown in Figure 6.



(a)
Guohui 国会/NN
Zongtong 总统/NN
Yu 与/CC
(b)
Guohui 国会/NN
Zongtong 总统
Rule: (与) 国会 → and parliment
and parliament
Rule: (总统) x1:NN → presidential x1
(c)
presidential and parliament

Figure 5: An example of translating a large HD fragment with the help of translations of its decomposed fragments.

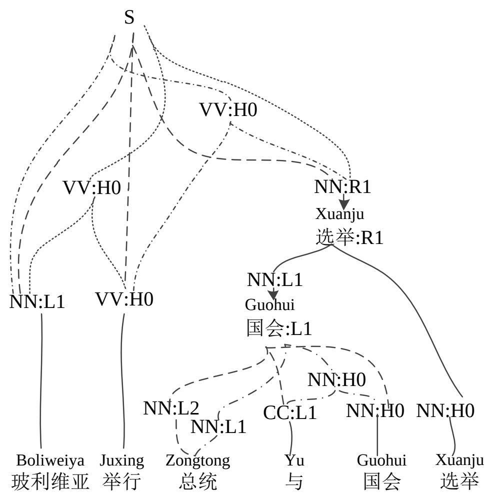


Figure 6: An example of a pseudo-forest for the dependency tree in Figure 1. It is represented using the constituent-tree style described in Section 3. Edges drawn in the same type of line are owned by the same sub-tree. Solid lines are shared edges.

In the pseudo-forest, we actually only create a forest structure for each HD fragment. For example, based on Figure 5, we create a constituent node labelled with “NN:H0” that covers the sub-fragment “$\overset{\text{Yu}}{\text{(与)}}$ $\overset{\text{Guohui}}{\text{国会}}$”. In so doing, a new node labelled with “NN:L1” is also created, which covers the Node “ $\overset{\text{Zongtong}}{\text{总统}}$ ”, because it is now the first left dependent in the sub-fragment “$\overset{\text{Zongtong}}{\text{(总统)}}$ $\overset{\text{Guohui}}{\text{国会}}$ ”.

Compared to the forest-based model (Mi et al., 2008), such a pseudo-forest cannot efficiently reduce the influence of parsing errors, but it is easily available and compatible with the Dep2Str Model.

corpus	sentences	words(ch)	words(en)
train	1,501,652	38,388,118	44,901,788
dev	878	22,655	26,905
MT04	1,597	43,719	52,705
MT05	1,082	29,880	35,326

Table 1: Chinese–English corpus. For the English dev and test sets, words counts are averaged across 4 references.

corpus	sentences	words(de)	words(en)
train	2,037,209	52,671,991	55,023,999
dev	3,003	72,661	74,753
test12	3,003	72,603	72,988
test13	3,000	63,412	64,810

Table 2: German–English corpus. In the dev and test sets, there is only one English reference for each German sentence.

5 Experiments

We conduct large-scale experiments to examine our methods on the Chinese–English and German–English translation tasks.

5.1 Data

The Chinese–English training corpus is from the LDC data, including LDC2002E18, LDC2003E07, LDC2003E14, LDC2004T07, the Hansards portion of LDC2004T08 and LDC2005T06. We take NIST 2002 as the development set to tune weights, and NIST 2004 (MT04) and NIST 2005 (MT05) as the test data to evaluate the systems. Table 1 provides a summary of the Chinese–English corpus.

The German–English training corpus is from WMT 2014, including Europarl V7 and News Commentary. News-test 2011 is taken as the development set, while News-test 2012 (test12) and News-test 2013 (test13) are our test sets. Table 2 provides a summary of the German–English corpus.

5.2 Baseline

For both language pairs, we filter sentence pairs longer than 80 words and keep the length ratio less than or equal to 3. English sentences are tokenized with scripts in Moses. Word alignment is performed by GIZA++ (Och and Ney, 2003) with the heuristic function *grow-diag-final-and* (Koehn et al., 2003). We use SRILM (Stolcke, 2002) to train a 5-gram language model on the Xinhua portion of the English Gigaword corpus 5th edition with modified Kneser-Ney discounting (Chen and Goodman, 1996). Minimum Error Rate Training (Och, 2003) is used to tune weights. Case-insensitive BLEU (Papineni et al., 2002) is used to evaluate the translation results. Bootstrap resampling (Koehn, 2004) is also performed to compute statistical significance with 1000 iterations.

Systems	MT05
XJ	33.91
D2S	33.79

Table 3: BLEU score [%] of the Dep2Str model before (**XJ**) and after (**D2S**) dependency tree being transformed. Systems are trained on a selected 1.2M Chinese–English corpus.

We implement the baseline Dep2Str model in Moses with methods described in this paper, which is denoted as **D2S**. The first experiment we do is to sanity check our implementation. Thus we take a separate system (denoted as **XJ**) for comparison which implements the Dep2Str model based on (Xie et al., 2011). As shown in Table 3, using the transformation of dependency trees, the Dep2Str model implemented in Moses (D2S) is comparable with the standard implementation (XJ).

In the rest of this section, we describe experiments which compare our system with Moses HPB (default setting), and test whether our decomposition approach improves performance over the baseline D2S.

As described in Section 2, the Dep2Str model only extracts phrase rules for translating a source word (head rule). This model could be enhanced by including phrase rules that cover more than one source word. Thus we also conduct experiments where phrase pairs[5] are added into our system. We set the length limit for phrase 7.

5.3 Chinese–English

In the Chinese–English translation task, the Stanford Chinese word segmenter (Chang et al., 2008) is used to segment Chinese sentences into words. The Stanford dependency parser (Chang et al., 2009) parses a Chinese sentence into the projective dependency tree.

[5]In this paper, the use of phrasal rules is similar to that of the HPB model, so they can be handled by Moses directly.

Systems	MT04	MT05
Moses HPB	35.56	33.99
D2S	33.93	32.56
+pseudo-forest	**34.28**	**34.10**
+sub-structural rules	**34.78**	**33.63**
+pseudo-forest	**35.46**	**34.13**
+phrase	**36.76***	**34.67***

Table 4: BLEU score [%] of our method and Moses HPB on the Chinese–English task. We use bold font to indicate that the result of our method is significantly better than D2S at $p \leq 0.01$ level, and * to indicate the result is significantly better than Moses HPB at $p \leq 0.01$ level.

Table 4 shows the translation results. We find that the decomposition approach proposed in this paper, including sub-structural rules and pseudo-forest, improves the baseline system D2S significantly (absolute improvement of +1.53/+1.57 (4.5%/4.8%, relative)). As a result, our system achieves comparable (-0.1/+0.14) results with Moses HPB. After including phrasal rules, our system performs significantly better (absolute improvement of +1.2/+0.68 (3.4%/2.0%, relative)) than Moses HPB on both test sets.[6]

5.4 German–English

We tokenize German sentences with scripts in Moses and use mate-tools[7] to perform morphological analysis and parse the sentence (Bohnet, 2010). Then the MaltParser[8] converts the parse result into the projective dependency tree (Nivre and Nilsson, 2005).

Experimental results in Table 5 show that incorporating sub-structural rules improves the baseline D2S system significantly (absolute improvement of +0.47/+0.63, (2.3%/2.8%, relative)), and achieves a slightly better (+0.08) result on test12 than Moses HPB. However, in the German–English task, the pseudo-forest produces a negative effect on the baseline system (-0.07/-0.45), despite the fact that our system combining both methods together is still better (+0.2/+0.11) than the baseline D2S. In the end, by resorting to phrasal rules, our system achieves the best performance overall which is significantly better (absolute improvement of +0.47/+0.59 (2.3%/2.6%, relative)) than Moses HPB.

[6]In our preliminary experiments, phrasal rules are also able to significantly improve our system on their own on both Chinese–English and German–English tasks, but the best performance is achieved by combining them with sub-structural rules and/or pseudo-forest.

[7]`http://code.google.com/p/mate-tools/`

[8]`http://www.maltparser.org/`

Systems	test12	test13
Moses HPB	20.44	22.77
D2S	20.05	22.13
+pseudo-forest	19.98	21.68
+sub-structural rules	**20.52**	**22.76**
+phrase	**20.91***	**23.46***
+pseudo-forest	20.25	22.24
+phrase	**20.75***	**23.20***

Table 5: BLEU score [%] of our method and Moses HPB on German–English task. We use bold font to indicate that the result of our method is significantly better than baseline D2S at $p \leq 0.01$ level, and * to indicate the result is significantly better than Moses HPB at $p \leq 0.01$ level.

Systems	# Rules	
	CE task	DE task
Moses HPB	388M	684M
D2S	27M	41M
+sub-structural rules	116M	121M
+phrase	215M	274M

Table 6: The number of rules in different systems On the Chinese–English (CE) and German–English (DE) corpus. Note that pseudo-forest (not listed) does not influence the number of rules.

5.5 Discussion

Besides long-distance reordering (Xie et al., 2011), another attraction of the Dep2Str model is its simplicity. It can perform fast translation with fewer rules than HPB. Table 6 shows the number of rules in each system. It is easy to see that all of our systems use fewer rules than HPB. However, the number of rules is not proportional to translation quality, as shown in Tables 4 and 5.

Experiments on the Chinese–English corpus show that it is feasible to translate the dependency tree via transformation for the Dep2Str model described in Section 2. Such a transformation causes the model to be easily integrated into Moses without making changes to the decoder, while at the same time producing comparable results with the standard implementation (shown in Table 3).

The decomposition approach proposed in this paper also shows a positive effect on the baseline Dep2Str system. Especially, sub-structural rules significantly improve the Dep2Str model on both Chinese–English and German–English tasks. However, experiments show that the pseudo-forest significantly improves the D2S system on the Chinese–English data, while it causes translation quality to decline on the German–English data.

Since using the pseudo-forest in our system is aimed at translating larger HD fragments via splitting it into pieces, we hypothesize that when translating German sentences, the pseudo-forest approach more likely results in much worse rules being applied. This is probably due to the shorter Mean Dependency Distance (MDD) and freer word order of German sentences(Eppler, 2013).

6 Conclusion

In this paper, we present an open source module which integrates a dependency-to-string model into Moses.

This module transforms an input dependency tree into a corresponding constituent tree during decoding which makes Moses perform dependency-based translation without necessitating any changes to the decoder. Experiments on Chinese–English show that the performance if our system is comparable with that of the standard dependency-based decoder.

Furthermore, we enhance the model by decomposing head-dependent fragments into smaller pieces. This decomposition enriches the Dep2Str model with more rules during training and allows us to create a pseudo-forest as input instead of a dependency tree during decoding. Large-scale experiments on Chinese–English and German–English tasks show that this decomposition can significantly improve the baseline dependency-to-string model on both language pairs. On the German–English task, sub-structural rules are more useful than the pseudo-forest input. In the end, by resorting to phrasal rules, our system performs significantly better than the hierarchical phrase-based model in Moses.

Our implementation of the dependency-to-string model with methods described in this paper is available at `http://computing.dcu.ie/~liangyouli/dep2str.zip`. In the future, we would like to conduct more experiments on other language pairs to examine this model, as well as reducing the restrictions on decomposition.

Acknowledgments

This research has received funding from the People Programme (Marie Curie Actions) of the European Union's Seventh Framework Programme FP7/2007-2013/ under REA grant agreement no. 317471. This research is also supported by the Science Foundation Ireland (Grant 12/CE/I2267) as part of the Centre for Next Generation Localisation at Dublin City University. The authors of this paper also thank the reviewers for helping to improve this paper.

References

Bernd Bohnet. 2010. Very High Accuracy and Fast Dependency Parsing is Not a Contradiction. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 89–97, Beijing, China.

Pi-Chuan Chang, Michel Galley, and Christopher D. Manning. 2008. Optimizing Chinese Word Segmentation for Machine Translation Performance. In *Proceedings of the Third Workshop on Statistical Machine Translation*, pages 224–232, Columbus, Ohio.

Pi-Chuan Chang, Huihsin Tseng, Dan Jurafsky, and Christopher D. Manning. 2009. Discriminative Reordering with Chinese Grammatical Relations Features. In *Proceedings of the Third Workshop on Syntax and Structure in Statistical Translation*, pages 51–59, Boulder, Colorado.

Stanley F. Chen and Joshua Goodman. 1996. An Empirical Study of Smoothing Techniques for Language Modeling. In *Proceedings of the 34th Annual Meeting on Association for Computational Linguistics*, pages 310–318, Santa Cruz, California.

John Cocke and Jacob T. Schwartz. 1970. Programming Languages and Their Compilers: Preliminary Notes. Technical report, Courant Institute of Mathematical Sciences, New York University, New York, NY.

Michael Collins, Lance Ramshaw, Jan Hajič, and Christoph Tillmann. 1999. A Statistical Parser for Czech. In *Proceedings of the 37th Annual Meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 505–512, College Park, Maryland.

Eva M. Duran Eppler. 2013. Dependency Distance and Bilingual Language Use: Evidence from German/English and Chinese/English Data. In *Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013)*, pages 78–87, Prague, August.

Heidi J. Fox. 2002. Phrasal Cohesion and Statistical Machine Translation. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing - Volume 10*, pages 304–3111, Philadelphia.

Liang Huang, Kevin Knight, and Aravind Joshi. 2006. A Syntax-directed Translator with Extended Domain of Locality. In *Proceedings of the Workshop on Computationally Hard Problems and Joint Inference in Speech and Language Processing*, pages 1–8, New York City, New York.

Richard Hudson. 1990. *English Word Grammar*. Blackwell, Oxford, UK.

Tadao Kasami. 1965. An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. Technical report, Air Force Cambridge Research Lab, Bedford, MA.

Philipp Koehn. 2004. Statistical Significance Tests for Machine Translation Evaluation. In *Proceedings of EMNLP 2004*, pages 388–395, Barcelona, Spain, July.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the 45th Annual Meeting of the ACL on Interactive Poster and Demonstration Sessions*, pages 177–180, Prague, Czech Republic.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, pages 48–54, Edmonton, Canada.

Arul Menezes and Chris Quirk. 2005. Dependency Treelet Translation: The Convergence of Statistical and Example-Based Machine-translation? In *Proceedings of the Workshop on Example-based Machine Translation at MT Summit X*, September.

Fandong Meng, Jun Xie, Linfeng Song, Yajuan Lü, and Qun Liu. 2013. Translation with Source Constituency and Dependency Trees. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1066–1076, Seattle, Washington, USA, October.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-Based Translation. In *Proceedings of ACL-08: HLT*, pages 192–199, June.

Joakim Nivre and Jens Nilsson. 2005. Pseudo-Projective Dependency Parsing. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 99–106, Ann Arbor, Michigan.

Franz Josef Och. 2003. Minimum Error Rate Training in Statistical Machine Translation. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, pages 160–167, Sapporo, Japan.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–51, March.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency Treelet Translation: Syntactically Informed Phrasal SMT. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 271–279, Ann Arbor, Michigan, June.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2010. String-to-Dependency Statistical Machine Translation. *Computational Linguistics*, 36(4):649–671, December.

Andreas Stolcke. 2002. SRILM-an Extensible Language Modeling Toolkit. In *Proceedings International Conference on Spoken Language Processing*, pages 257–286, November.

Fei Xia and Martha Palmer. 2001. Converting Dependency Structures to Phrase Structures. In *Proceedings of the First International Conference on Human Language Technology Research*, pages 1–5, San Diego.

Jun Xie, Haitao Mi, and Qun Liu. 2011. A Novel Dependency-to-string Model for Statistical Machine Translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 216–226, Edinburgh, United Kingdom.

Jun Xie, Jinan Xu, and Qun Liu. 2014. Augment Dependency-to-String Translation with Fixed and Floating Structures. In *Proceedings of the 25th International Conference on Computational Linguistics*, pages 2217–2226, Dublin, Ireland.

Deyi Xiong, Qun Liu, and Shouxun Lin. 2007. A Dependency Treelet String Correspondence Model for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 40–47, Prague, June.

Daniel H. Younger. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10(2):189–208.

Arnold M. Zwicky. 1985. Heads. *Journal of Linguistics*, 21:1–29, 3.