Statistical Syntax-Directed Translation with Extended Domain of Locality

Liang Huang
University of Pennsylvania
Philadelphia, PA 19104
lhuang3@cis.upenn.edu

Kevin Knight
USC/ISI
Marina del Rey, CA 90292
knight@isi.edu

Aravind Joshi
University of Pennsylvania
Philadelphia, PA 19104
joshi@cis.upenn.edu

Abstract

In syntax-directed translation, the source-language input is first parsed into a parse-tree, which is then recursively converted into a string in the target-language. We model this conversion by an extended tree-to-string transducer that has multi-level trees on the source-side, which gives our system more expressive power and flexibility. We also define a direct probability model and use a linear-time dynamic programming algorithm to search for the best derivation. The model is then extended to the general log-linear framework in order to incorporate other features like n-gram language models. We devise a simple-yet-effective algorithm to generate non-duplicate k-best translations for n-gram rescoring. Preliminary experiments on English-to-Chinese translation show a significant improvement in terms of translation quality compared to a state-of-the-art phrase-based system.

1 Introduction

The concept of *syntax-directed translation* was originally proposed in compiling (Irons, 1961; Lewis and Stearns, 1968; Aho and Ullman, 1972), where the source program is parsed into a tree representation that guides the generation of the object code. In other words, the translation is directed by a syntactic tree. In this context, a syntax-directed translator consists of two components, a source-language parser and a recursive converter which is usually modeled as a top-down tree-to-string transducer (Gécseg and Steinby, 1984).

This paper adapts the idea of syntax-directed translation to statistical machine translation (MT). We apply stochastic operations at each node of the source-language parse-tree and search for the best derivation (a sequence of translation steps) that converts the whole tree into some target-language string with the highest probability. However, the structural divergence across languages often results in non-isomorphic parse-trees that is beyond the power of SCFGs. For example, the *S(VO)* structure in English is translated into a *VSO* word-order in Arabic, an instance of *complex reordering* not captured by any SCFG (Fig. 1).

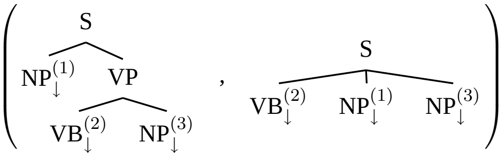


Figure 1: An example of complex reordering represented as an STSG rule, which is beyond any SCFG.

To alleviate the non-isomorphism problem, (synchronous) grammars with richer expressive power have been proposed whose rules apply to larger fragments of the tree. For example, Shieber and Schabes (1990) introduce synchronous tree-adjoining grammar (STAG) and Eisner (2003) uses a synchronous tree-substitution grammar (STSG), which is a restricted version of STAG with no adjunctions. STSGs and STAGs generate more *tree relations* than SCFGs, e.g. the non-isomorphic tree pair in Fig. 1. This extra expressive power lies in the *extended domain of locality* (EDL) (Joshi and Schabes, 1997), i.e., elementary structures beyond the scope of one-level context-free productions. Besides being linguistically motivated, the need for EDL is also supported by empirical findings in MT that one-level rules are often inadequate (Fox, 2002; Galley et al., 2004). Similarly, in the tree-transducer terminology, Graehl and Knight (2004) define extended tree transducers that have multi-level trees on the source-side.

Since syntax-directed translation models sep-

Proceedings of the 7th Conference of the Association for Machine Translation in the Americas, pages 66-73,
Cambridge, August 2006. ©2006 The Association for Machine Translation in the Americas

arate the source-language analysis from the recursive transformation, the domains of locality in these two modules are orthogonal to each other: in this work, we use a CFG-based Treebank parser but focus on the extended domain in the recursive converter. Following Galley et al. (2004), we use a special class of *extended tree-to-string transducer* (**xRs** for short) with multi-level left-hand-side (LHS) trees.[1] Since the right-hand-side (RHS) string can be viewed as a flat one-level tree with the same nonterminal root from LHS (Fig. 1), this framework is closely related to STSGs: they both have extended domain of locality on the source-side, while our framework remains as a CFG on the target-side. For instance, an equivalent **xRs** rule for the complex reordering in Fig. 1 would be

$$\text{S}(x_1\text{:NP, VP}(x_2\text{:VB, }x_3\text{:NP})) \rightarrow x_2\ x_1\ x_3$$

While Section 3 will define the model formally, we first proceed with an example translation from English to Chinese (note in particular the inverted phrases between source and target):

(1) the gunman was killed by the police .

qiangshou bei *jingfang jibi* 。
[gunman] [passive] [police] [killed] .

Figure 2 shows how the system works. The English sentence (a) is first parsed into the tree in (b), which is then recursively converted into the Chinese string in (e) through five steps. First, at the root node, we apply the rule r_1 which preserves the top-level word-order and translates the English period into its Chinese counterpart:

(r_1) S (x_1:NP-C x_2:VP PUNC (.)) → x_1 x_2 。

Then, the rule r_2 grabs the whole sub-tree for "the gunman" and translates it as a phrase:

(r_2) NP-C (DT (the) NN (gunman)) → *qiangshou*

Now we get a "partial Chinese, partial English" sentence "*qiangshou* VP 。" as shown in Fig. 2 (c). Our recursion goes on to translate the VP sub-tree. Here we use the rule r_3 for the passive construction:

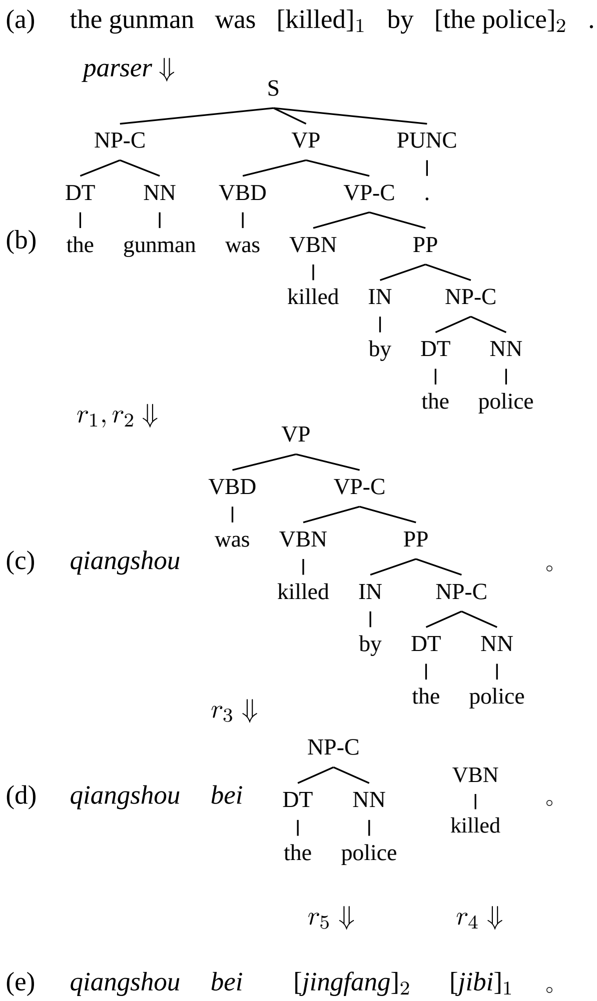
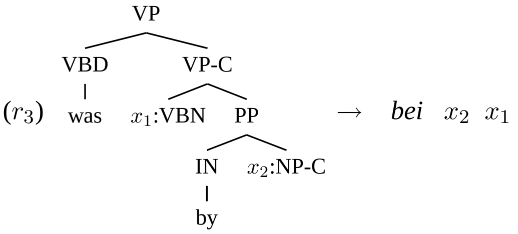


Figure 2: A synatx-directed translation process for Example (1).



which captures the fact that the agent (NP-C, "the police") and the verb (VBN, "killed") are always inverted between English and Chinese in a passive voice. Finally, we apply rules r_4 and r_5 which perform phrasal translations for the two remaining sub-trees in (d), respectively, and get the completed Chinese string in (e).

[1] Throughout this paper, we will use LHS and source-side interchangeably (similarly, RHS and target-side). In accordance with our experiments, we also use English and Chinese as the source and target languages, opposite the Foreign-to-English convention of Brown et al. (1993).

2 Previous Work

It is helpful to compare this approach with recent efforts in statistical MT. Phrase-based models (Koehn et al., 2003; Och and Ney, 2004) are good at learning local translations that are pairs of (consecutive) sub-strings, but often insufficient in modeling the reorderings of phrases themselves, especially between language pairs with very different word-order. This is because the generative capacity of these models lies within the realm of finite-state machinery (Kumar and Byrne, 2003), which is unable to process nested structures and long-distance dependencies in natural languages.

Syntax-based models aim to alleviate this problem by exploiting the power of synchronous rewriting systems. Both Yamada and Knight (2001) and Chiang (2005) use SCFGs as the underlying model and do parsing and transformation in a joint search, essentially over a packed forest of parse-trees. To this end, their methods are not *directed* by a syntactic tree. Although their method potentially considers more than one single parse-tree as in our case, the packed representation of the forest restricts the scope of each transfer step to a one-level context-free rule, while our approach decouples the source-language analyzer and the recursive converter, so that the latter can have an extended domain of locality. In addition, our model also enjoys a speed-up by this decoupling, with each of the two stages having a smaller search space. In fact, the recursive transfer step can be done by a *linear-time* algorithm (see Section 5), and the parsing step is also fast with the modern Treebank parsers, for instance (Collins, 1999; Charniak, 2000). In contrast, their decodings are reported to be computationally expensive and Chiang (2005) uses aggressive pruning to make it tractable. There also exists a compromise between these two approaches, which uses a k-best list of parse trees (for a relatively small k) to approximate the full forest (see future work).

Our model, being linguistically motivated, is also more expressive than the formally syntax-based models of Chiang (2005) and Wu (1997). Consider, again, the passive example in rule r_3. In Chiang's SCFG, there is only one nonterminal X, so a corresponding rule would be

$$\langle \text{ was X}^{(1)} \text{ by X}^{(2)},\ \textit{bei}\ \text{X}^{(2)}\ \text{X}^{(1)} \rangle$$

which can also pattern-match the English sentence:

$$\text{I was [asleep]}_1 \text{ by [sunset]}_2 \ .$$

and translate it into Chinese as a passive voice. This produces very odd Chinese translation, because here "was A by B" in the English sentence is *not* a passive construction. By contrast, our model applies rule r_3 only if A is a past participle (VBN) and B is a noun phrase (NP-C). This example also shows that, one-level SCFG rule, even if informed by the Treebank as in (Yamada and Knight, 2001), is not enough to capture a common construction like this which is five levels deep (from VP to "by").

Recent works on dependency-based MT (Lin, 2004; Ding and Palmer, 2005; Quirk et al., 2005) are closest to this work in the sense that their translations are also based on source-language parse trees. The difference is that they use dependency trees instead of constituent trees. Although they share with this work the basic motivations and similar speed-up, it is difficult to specify reordering information within dependency elementary structures, so they either resort to heuristics (Lin) or a separate ordering model for linearization (the other two works). Our approach, in contrast, explicitly models the re-ordering of sub-trees within individual transfer rules. In addition, it is more appropriate to call their models "(lightweight-) semantics-directed" since dependency structure is closer to the semantic representation.

3 Extended Tree-to-String Tranducers

In this section, we define the formal machinery of our recursive transformation model as a special case of **xRs** transducers (Graehl and Knight, 2004) that has only one state, and each rule is linear (L) and non-deleting (N) with regards to variables in the source and target sides (hence the name 1-**xRLNs**).

Definition 1. A **1-xRLNs transducer** is a tuple $(N, \Sigma, \Delta, \mathcal{R})$ where N is the set of nonterminals, Σ is the input alphabet, Δ is the output alphabet, and $\mathcal{R}$ is a set of rules. A rule in $\mathcal{R}$ is a tuple (t, s, ϕ) where:

1. t is the LHS tree, whose internal nodes are labeled by nonterminal symbols, and whose frontier nodes are labeled terminals from Σ or variables from a set $\mathcal{X} = \{x_1, x_2, \ldots\}$;
2. $s \in (\mathcal{X} \cup \Delta)^*$ is the RHS string;

3. ϕ is a mapping from $\mathcal{X}$ to nonterminals N.

We require each variable $x_i \in \mathcal{X}$ occurs *exactly once* in t and *exactly once* in s (linear and non-deleting).

We denote $\rho(t)$ to be the **root symbol** of tree t. When writing these rules, we avoid notational overhead by introducing a short-hand form from Galley et al. (2004) that integrates the mapping into the tree, which is used throughout Section 1. Following TSG terminology (see Figure 1), we call these "variable nodes" such as x_2:NP-C *substitution nodes*, since when applying a rule to a tree, these nodes will be matched with a sub-tree with the same root symbol.

We also define $|\mathcal{X}|$ to be the **rank** of the rule, i.e., the number of variables in it. For example, rules r_1 and r_3 in Section 1 are both of rank 2. If a rule has no variable, i.e., it is of rank zero, then it is called a *purely lexical rule*, which performs a phrasal translation as in phrase-based models. Rule r_2, for instance, can be thought of as a phrase pair $\langle$the gunman, *qiangshou*$\rangle$.

Informally speaking, a derivation in a transducer is a sequence of steps converting a source-language tree into a target-language string, with each step applying one tranduction rule. However, it can also be formalized as a tree, following the notion of *derivation-tree* in TAG (Joshi and Schabes, 1997):

Definition 2. A **derivation** d, its **source and target projections**, noted $\mathcal{E}(d)$ and $\mathcal{C}(d)$ respectively, are recursively defined as follows:

1. If $r = (t, s, \phi)$ is a purely lexical rule ($\phi = \emptyset$), then $d = r$ is a derivation, where $\mathcal{E}(d) = t$ and $\mathcal{C}(d) = s$;

2. If $r = (t, s, \phi)$ is a rule, and d_i is a (sub-) derivation with the root symbol of its source projection matches the corresponding substitution node in r, i.e., $\rho(\mathcal{E}(d_i)) = \phi(x_i)$, then $d = r(d_1, \ldots, d_m)$ is also a derivation, where $\mathcal{E}(d) = [x_i \mapsto \mathcal{E}(d_i)]t$ and $\mathcal{C}(d) = [x_i \mapsto \mathcal{C}(d_i)]s$.

Note that we use a short-hand notation $[x_i \mapsto y_i]t$ to denote the result of substituting each x_i with y_i in t, where x_i ranges over all variables in t.

For example, Figure 3 shows two derivations for the sentence pair in Example (1). In both cases, the source projection is the English tree in Figure 2 (b), and the target projection is the Chinese translation.

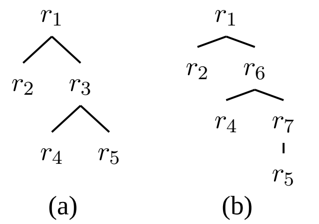


Figure 3: (a) the derivation in Figure 2; (b) another derviation producing the same output by replacing r_3 with r_6 and r_7, which provides another way of translating the passive construction:

(r_6) VP (VBD (was) VP-C (x_1:VBN x_2:PP)) → x_2 x_1
(r_7) PP (IN (by) x_1:NP-C) → *bei* x_1

Galley et al. (2004) presents a linear-time algorithm for automatic extraction of these **xRs** rules from a parallel corpora with word-alignment and parse-trees on the source-side, which will be used in our experiments in Section 6.

4 Probability Models

4.1 Direct Model

Departing from the conventional noisy-channel approach of Brown et al. (1993), our basic model is a *direct* one:

$$c^* = \operatorname*{argmax}_c \Pr(c \mid e) \tag{2}$$

where e is the English input string and c^* is the best Chinese translation according to the translation model $\Pr(c \mid e)$. We now marginalize over all English parse trees $\mathcal{T}(e)$ that yield the sentence e:

$$\begin{aligned} \Pr(c \mid e) &= \sum_{\tau \in \mathcal{T}(e)} \Pr(\tau, c \mid e) \\ &= \sum_{\tau \in \mathcal{T}(e)} \Pr(\tau \mid e) \Pr(c \mid \tau) \end{aligned} \tag{3}$$

Rather than taking the sum, we pick the best tree τ^* and factors the search into two separate steps: parsing (4) (a well-studied problem) and tree-to-string translation (5) (Section 5):

$$\tau^* = \operatorname*{argmax}_{\tau \in \mathcal{T}(e)} \Pr(\tau \mid e) \tag{4}$$

$$c^* = \operatorname*{argmax}_c \Pr(c \mid \tau^*) \tag{5}$$

In this sense, our approach can be considered as a Viterbi approximation of the computationally expensive joint search using (3) directly. Similarly, we now marginalize over all derivations

$$\mathcal{D}(\tau^*) = \{d \mid \mathcal{E}(d) = \tau^*\}$$

that translates English tree τ into some Chinese string and apply the Viterbi approximation again to search for the best derivation d^*:

$$c^* = \mathcal{C}(d^*) = \mathcal{C}(\operatorname*{argmax}_{d \in \mathcal{D}(\tau^*)} \Pr(d)) \tag{6}$$

Assuming different rules in a derivation are applied independently, we approximate $\Pr(d)$ as

$$\Pr(d) = \prod_{r \in d} \Pr(r) \tag{7}$$

where the probability $\Pr(r)$ of the rule r is estimated by conditioning on the root symbol $\rho(t(r))$:

$$\begin{aligned} \Pr(r) &= \Pr(t(r), s(r) \mid \rho(t(r))) \\ &= \frac{c(r)}{\sum_{r':\rho(t(r'))=\rho(t(r))} c(r')} \end{aligned} \tag{8}$$

where $c(r)$ is the count (or frequency) of rule r in the training data.

4.2 Log-Linear Model

Following Och and Ney (2002), we extend the direct model into a general log-linear framework in order to incorporate other features:

$$c^* = \operatorname*{argmax}_{c} \Pr(c \mid e)^\alpha \cdot \Pr(c)^\beta \cdot e^{-\lambda|c|} \tag{9}$$

where $\Pr(c)$ is the language model and $e^{-\lambda|c|}$ is the length penalty term based on $|c|$, the length of the translation. Parameters α, β, and λ are the weights of relevant features. Note that positive λ prefers longer translations, thus we call λ the length-bonus parameter. We use a standard trigram model for $\Pr(c)$.

5 Search Algorithms

We first present a linear-time algorithm for searching the best derivation under the direct model, and then extend it to the log-linear case by a new variant of k-best parsing.

5.1 Direct Model: Memoized Recursion

Since our probability model is not based on the noisy channel, we do not call our search module a "decoder" as in most statistical MT work. Instead, readers who speak English but not Chinese can view it as an "encoder" (or encryptor), which corresponds exactly to our *direct* model.

Given a fixed parse-tree τ^*, we are to search for the best derivation with the highest probability. This can be done by a simple top-down traversal (or depth-first search) from the root of τ^*: at each node η in τ^*, try each possible rule r whose English-side pattern $t(r)$ matches the subtree τ^*_η rooted at η, and recursively visit each descendant node η_i in τ^*_η that corresponds to a variable in $t(r)$. We then collect the resulting target-language strings and plug them into the Chinese-side $s(r)$ of rule r, getting a translation for the subtree τ^*_η. We finally take the best of all translations.

With the extended LHS of our transducer, there may be many different rules applicable at one tree node. For example, consider the VP subtree in Fig. 2 (c), where both r_3 and r_6 can apply. As a result, the number of derivations is exponential in the size of the tree, since there are exponentially many decompositions of the tree for a given set of rules. This problem can be solved by *memoization* (Cormen et al., 2001): we cache each subtree that has been visited before, so that every tree node is visited *at most* once. This results in a dynamic programming algorithm that is guaranteed to run in $O(npq)$ time where n is the size of the parse tree, p is the maximum number of rules applicable to one tree node, and q is the maximum size of an applicable rule. For a given rule-set, this algorithm runs in time linear to the length of the input sentence, since p and q are considered grammar constants, and n is proportional to the input length. The full pseudo-code is worked out in Algorithm 1. A restricted version of this algorithm first appears in compiling for optimal code generation from expression-trees (Aho and Johnson, 1976). In computational linguistics, the bottom-up version of this algorithm resembles the *tree parsing* algorithm for TSG by Eisner (2003). Similar algorithms have also been proposed for dependency-based translation (Lin, 2004; Ding and Palmer, 2005).

5.2 Log-linear Model: rescoring non-duplicate k-best translations

Under the log-linear model, one still prefers to search for the globally best derivation d^*:

$$d^* = \operatorname*{argmax}_{d \in \mathcal{D}(\tau^*)} \Pr(d)^\alpha \Pr(\mathcal{C}(d))^\beta e^{-\lambda|\mathcal{C}(d)|} \tag{10}$$

Algorithm 1 Top-down Memoized Recursion

```
function TRANSLATE($\eta$)
    if $cache[\eta]$ defined then                                  ▷ this sub-tree visited before?
        return $cache[\eta]$
    $best \leftarrow 0$
    for $r \in \mathcal{R}$ do                                          ▷ try each rule $r$
        $matched,\ sublist \leftarrow$ PATTERNMATCH($t(r), \eta$)            ▷ tree pattern matching
        if $matched$ then      ▷ if matched, $sublist$ contains a list of matched subtrees
            $prob \leftarrow \Pr(r)$                                 ▷ the probability of rule $r$
            for $\eta_i \in sublist$ do
                $p_i, s_i \leftarrow$ TRANSLATE($\eta_i$)          ▷ recursively solve each sub-problem
                $prob \leftarrow prob \cdot p_i$
            if $prob > best$ then
                $best \leftarrow prob$
                $str \leftarrow [x_i \mapsto s_i]s(r)$                         ▷ plug in the results
    $cache[\eta] \leftarrow best,\ str$                  ▷ caching the best solution for future use
    return $cache[\eta]$                            ▷ returns the best string with its prob.
```

However, integrating the n-gram model $\Pr(\mathcal{C}(d))$ with the translation model in the search is computationally very expensive. As a standard alternative, rather than aiming at the exact best derivation, we search for top-k derivations under the direct model using Algorithm 1, and then rerank the k-best list with the language model and length penalty.

Like other instances of dynamic programming, Algorithm 1 can be viewed as a hypergraph search problem. To this end, we use an efficient algorithm by Huang and Chiang (2005, Algorithm 3) that solves the general k-best derivations problem in monotonic hypergraphs. It consists of a normal forward phase for the 1-best derivation and a recursive backward phase for the 2nd, 3rd, $\ldots$, k^{th} derivations.

Unfortunately, different derivations may have the same yield (a problem called *spurious ambiguity*), due to multi-level LHS of our rules. In practice, this results in a very small ratio of unique strings among top-k derivations, while the rescoring approach prefers diversity within the k-best list. To alleviate this problem, determinization techniques have been proposed by Mohri and Riley (2002) for finite-state automata and extended to tree automata by May and Knight (2006). These methods eliminate spurious ambiguity by effectively transforming the grammar into an equivalent deterministic form. However, this transformation often leads to a blow-up in forest size, which is exponential in the original size in the worst-case.

So instead of determinization, here we present a simple-yet-effective extension to the Algorithm 3 of Huang and Chiang (2005) that guarantees to output unique translated strings:

- keep a hash-table of unique strings at each vertex in the hypergraph
- when asking for the next-best derivation of a vertex, keep asking until we get a new string, and then add it into the hash-table

This method should work in general for any equivalence relation (say, same derived tree) that can be defined on derivations.

6 Experiments

Our experiments are on English-to-Chinese translation, the opposite direction to most of the recent work in SMT. We are not doing the reverse direction at this time partly due to the lack of a sufficiently good parser for Chinese.

6.1 Data Preparation

Our training set is a Chinese-English parallel corpus with 1.95M aligned sentences (28.3M words on the English side). We first word-align them by GIZA++, then parse the English side by a variant of Collins (1999) parser, and finally apply the rule-extraction algorithm of Galley et al. (2004). The resulting rule set has 24.7M **xRs** rules. We also use the SRI Language Modeling Toolkit (Stolcke, 2002) to train a Chinese trigram model with Knesser-Ney smoothing on the Chinese side of the parallel corpus.

Table 1: BLEU score results on dev set and test set (1-reference, character-based)

	dev set	**test set** (140 sentences)	
System	**BLEU-4**	**BLEU-4**	**BLEU-8**
Pharaoh (with max-BLEU tuning)	25.96 ±2.8	23.54 ±1.9	6.739 ±1.2
direct model (1-best)	22.10 ±2.6	24.53 ±2.2	7.309 ±1.9
log-linear model (rescoring non-duplicate k-best list)			
$k = 5000$ $(\beta = 0.994, \lambda = 0.513)$	26.01 ±2.7	25.74 ±2.3	8.489 ±2.1
$k = 50000$ $(\beta = 0.793, \lambda = 0.469)$	26.95 ±2.8	26.69 ±2.4	9.323 ±2.2

Our evaluation data is constructed by inverting the direction of NIST evaluation data as follows: we take the 140 short sentences with less than 25 Chinese words from the Xinhua portion of the NIST 2003 Chinese-to-English evaluation set, and divide them into dev-set and test-set, each with 70 Chinese sentences. In both sets, we use the first and second English references of each Chinese sentence as our source input, and the original Chinese sentences as our single reference. So we end up with 140 English sentences to translate in both dev and test sets. Note that this arrangement makes sure the test set is blind.

6.2 Systems

We implemented our system as follows: for each input sentence, we first run Algorithm 1, which returns the 1-best translation and also builds the derivation forest of all translations for this sentence. Then we extract the top-k non-duplicate translated strings from this forest using the algorithm in Section 5.2 and rescore them with the trigram model and the length penalty.

We compared our system with a state-of-the-art phrase-based system Pharaoh (Koehn, 2004) on the evaluation data. Since the target language is Chinese, we will report character-based BLEU scores instead of word-based to ensure our results are independent of Chinese tokenizations (although our language models are word-based). Feature weights of both systems are tuned for BLEU-4 (up to 4-grams) on the dev set. For Pharaoh, we use the standard minimum error-rate training (Och, 2003) (David Chiang's implementation); and for our system, since there are only two independent features (as we always fix $\alpha = 1$), we use a simple grid-based line-optimization along the language-model weight axis. For a given language-model weight β, we use binary search to find the best length bonus parameter λ that leads to a length-ratio closest to 1 against the reference.

6.3 Results and Statistical Significance

The BLEU scores with 95% confidence intervals are presented in Table 1. On both development set and test set, rescored translations are significantly better than the 1-best results from the direct model, and the larger k is, the better the result after rescoring. On the test set, our 50000-best rescoring result has a BLEU score of 26.69, which is significantly better than Pharaoh's 23.54 ($p < 0.05$, using the *sign-test* suggested by Collins et al. (2005)). The difference in BLEU-8 scores is also statistically significant ($p < 0.01$). These preliminary experiments show that our approach is very promising.

7 Conclusion and Future Work

This paper presents an adaptation of the classic syntax-directed translation with linguistically-motivated formalisms for statistical MT. Currently we are investigating efficient algorithms for principled integration of n-gram models in the search, rather than k-best rescoring. Besides, we will extend this work to translating the top k parse trees, instead of committing to the 1-best tree, as parsing errors affect translation quality (Quirk and Corston-Oliver, 2006).

Acknowledgements

The authors wish to thank Wei Wang, Radu Soricut and Steve Deneefe for help with data and tools. We are also grateful to David Chiang, Yuan Ding, Jonathan Graehl, Daniel Marcu, Mitch Marcus, Bob Moore, Fernando Pereira, Chris Quirk, Giorgio Satta and Hao Zhang for helpful discussions. This work was partially supported by NSF ITR grants EIA-0205456 and IIS-0428020.

References

A. V. Aho and S. C. Johnson. 1976. Optimal code generation for expression trees. *J. ACM*, 23(3):488–501.

Alfred V. Aho and Jeffrey D. Ullman. 1972. *The Theory of Parsing, Translation, and Compiling*, volume I: Parsing. Prentice Hall, Englewood Cliffs, New Jersey.

Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19:263–311.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proc. of NAACL*, pages 132–139.

David Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of the 43rd ACL*.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause restructuring for statistical machine translation. In *Proc. of ACL*.

Michael Collins. 1999. *Head-Driven Statistical Models for Natural Language Parsing*. Ph.D. thesis, University of Pennsylvania.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2001. *Introduction to Algorithms*. MIT Press, second edition.

Yuan Ding and Martha Palmer. 2005. Machine translation using probablisitic synchronous dependency insertion grammars. In *Proceedings of the 43rd ACL*.

Jason Eisner. 2003. Learning non-isomorphic tree mappings for machine translation. In *Proceedings of ACL (companion volume)*, pages 205–208.

Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *In Proc. of EMNLP*.

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

F. Gécseg and M. Steinby. 1984. *Tree Automata*. Akadémiai Kiadó, Budapest.

Jonathan Graehl and Kevin Knight. 2004. Training tree transducers. In *HLT-NAACL*, pages 105–112.

Liang Huang and David Chiang. 2005. Better k-best Parsing. In *Proceedings of the Nineth International Workshop on Parsing Technologies (IWPT-2005), 9-10 October 2005, Vancouver, Canada*.

E. T. Irons. 1961. A syntax-directed compiler for ALGOL 60. *Comm. ACM*, 4(1):51–55.

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

Philipp Koehn, Franz Joseph Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proc. of HLT-NAACL*, pages 127–133.

Philipp Koehn. 2004. Pharaoh: a beam search decoder for phrase-based statistical machine translation models. In *Proc. of AMTA*, pages 115–124.

Shankar Kumar and William Byrne. 2003. A weighted finite state transducer implementation of the alignment template model for statistical machine translation. In *Proc. of HLT-NAACL*, pages 142–149.

P. M. Lewis and R. E. Stearns. 1968. Syntax-directed transduction. *Journal of the ACM*, 15(3):465–488.

Dekang Lin. 2004. A path-based transfer model for machine translation. In *Proceedings of the 20th COLING*.

Jonathan May and Kevin Knight. 2006. A better n-best list: Practical determinization of weighted finite tree automata. In *Proc. of HLT-NAACL 2006*.

Mehryar Mohri and Michael Riley. 2002. An efficient algorithm for the n-best-strings problem. In *Proceedings of the International Conference on Spoken Language Processing 2002 (ICSLP '02)*, Denver, Colorado, September.

Franz Josef Och and Hermann Ney. 2002. Discriminative training and maximum entropy models for statistical machine translation. In *Proc. of ACL*.

F. J. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30:417–449.

Franz Och. 2003. Minimum error rate training for statistical machine translation. In *Proc. of ACL*.

Chris Quirk and Simon Corston-Oliver. 2006. The impact of parse quality on syntactically-informed statistical machine translation. In *Proc. of EMNLP*.

Chris Quirk, Arul Menezes, and Colin Cherry. 2005. Dependency treelet translation: Syntactically informed phrasal smt. In *Proceedings of the 43rd ACL*.

Stuart Shieber and Yves Schabes. 1990. Synchronous tree-adjoining grammars. In *Proc. of COLING*, pages 253–258.

Andrea Stolcke. 2002. Srilm: an extensible language modeling toolkit. In *Proc. of ICSLP*.

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

Kenji Yamada and Kevin Knight. 2001. A syntax-based statistical translation model. In *Proc. of ACL*.