Quadratic-Time Dependency Parsing for Machine Translation

Michel Galley
Computer Science Department
Stanford University
Stanford, CA 94305-9020
mgalley@cs.stanford.edu

Christopher D. Manning
Computer Science Department
Stanford University
Stanford, CA 94305-9010
manning@cs.stanford.edu

Abstract

Efficiency is a prime concern in syntactic MT decoding, yet significant developments in statistical parsing with respect to asymptotic efficiency haven't yet been explored in MT. Recently, McDonald et al. (2005b) formalized dependency parsing as a maximum spanning tree (MST) problem, which can be solved in quadratic time relative to the length of the sentence. They show that MST parsing is almost as accurate as cubic-time dependency parsing in the case of English, and that it is more accurate with free word order languages. This paper applies MST parsing to MT, and describes how it can be integrated into a phrase-based decoder to compute dependency language model scores. Our results show that augmenting a state-of-the-art phrase-based system with this dependency language model leads to significant improvements in TER (0.92%) and BLEU (0.45%) scores on five NIST Chinese-English evaluation test sets.

1 Introduction

Hierarchical approaches to machine translation have proven increasingly successful in recent years (Chiang, 2005; Marcu et al., 2006; Shen et al., 2008), and often outperform phrase-based systems (Och and Ney, 2004; Koehn et al., 2003) on target-language fluency and adequacy. However, their benefits generally come with high computational costs, particularly when chart parsing, such as CKY, is integrated with language models of high orders (Wu, 1996). Indeed, synchronous CFG parsing with m-grams runs in $O(n^{3m})$ time, where n is the length of the sentence.[1]

Furthermore, synchronous CFG approaches often only marginally outperform the most competitive phrase-based systems in large-scale experiments such as NIST evaluations.[2] This lack of significant difference may not be completely surprising. Indeed, researchers have shown that gigantic language models are key to state-of-the-art performance (Brants et al., 2007), and the ability of phrase-based decoders to handle large-size, high-order language models with no consequence on asymptotic running time during decoding presents a compelling advantage over CKY decoders, whose time complexity grows prohibitively large with higher-order language models.

While context-free decoding algorithms (CKY, Earley, etc.) may sometimes appear too computationally expensive for high-end statistical machine translation, there are many alternative parsing algorithms that have seldom been explored in the machine translation literature. The parsing literature presents faster alternatives for both phrase-structure and dependency trees, e.g., $O(n)$ shift-reduce parsers and variants ((Ratnaparkhi, 1997; Nivre, 2003), *inter alia*). While deterministic parsers are often deemed inadequate for dealing with ambiguities of natural language, highly accurate $O(n^2)$ algorithms exist in the case of dependency parsing. Building upon the theoretical work of (Chu and Liu, 1965; Edmonds, 1967), McDonald et al. (2005b) present a quadratic-time dependency parsing algorithm that is just 0.7% less accurate than "full-fledged" chart parsing (which, in the case of dependency parsing, runs in time $O(n^3)$ (Eisner, 1996)).

In this paper, we show how to exploit syntactic dependency structure for better machine translation, under the constraint that the depen-

[1] The algorithmic complexity of (Wu, 1996) is $O(n^{3+4(m-1)})$, though Huang et al. (2005) present a more efficient factorization inspired by (Eisner and Satta, 1999) that yields an overall complexity of $O(n^{3+3(m-1)})$, i.e., $O(n^{3m})$. In comparison, phrase-based decoding can run in linear time if a distortion limit is imposed. Of course, this comparison holds only for approximate algorithms. Since exact MT decoding is NP complete (Knight, 1999), there is no exact search algorithm for either phrase-based or syntactic MT that runs in polynomial time (unless P = NP).

[2] Results of the 2008 NIST Open MT evaluation (http://www.itl.nist.gov/iad/mig/tests/mt/2008/doc/mt08_official_results_v0.html) reveal that, while many of the best systems in the Chinese-English and Arabic-English tasks incorporate synchronous CFG models, score differences with the best phrase-based system were insignificantly small.

Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP, pages 773–781,
Suntec, Singapore, 2-7 August 2009.

dency structure is built as a by-product of phrase-based decoding, without reliance on a dynamic-programming or chart parsing algorithm such as CKY or Earley. Adapting the approach of McDonald et al. (2005b) for machine translation, we incrementally build dependency structure left-to-right in time $O(n^2)$ during decoding. Most interestingly, the time complexity of non-projective dependency parsing remains quadratic as the order of the language model increases. This provides a compelling advantage over previous dependency language models for MT (Shen et al., 2008), which use a 5-gram LM only during reranking. In our experiments, we build a competitive baseline (Koehn et al., 2007) incorporating a 5-gram LM trained on a large part of Gigaword and show that our dependency language model provides improvements on five different test sets, with an overall gain of 0.92 in TER and 0.45 in BLEU scores. These results are found to be statistically very significant ($p \leq .01$).

2 Dependency parsing for machine translation

In this section, we review dependency parsing formulated as a maximum spanning tree problem (McDonald et al., 2005b), which can be solved in quadratic time, and then present its adaptation and novel application to phrase-based decoding.

Dependency models have recently gained considerable interest in many NLP applications, including machine translation (Ding and Palmer, 2005; Quirk et al., 2005; Shen et al., 2008). Dependency structure provides several compelling advantages compared to other syntactic representations. First, dependency links are close to the semantic relationships, which are more likely to be consistent across languages. Indeed, Fox (2002) found inter-lingual phrasal cohesion to be greater than for a CFG when using a dependency representation, for which she found only 12.6% of head crossings and 9.2% modifier crossings. Second, dependency trees contain exactly one node per word, which contributes to cutting down the search space during parsing: indeed, the task of the parser is merely to connect existing nodes rather than hypothesizing new ones. Finally, dependency models are more flexible and account for (non-projective) head-modifier relations that CFG models fail to represent adequately, which is problematic with certain types of grammatical constructions and with free word order languages,

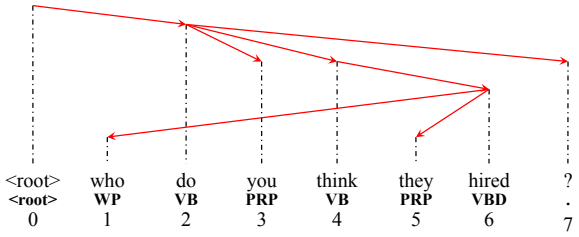

Figure 1: A dependency tree with directed edges going from heads to modifiers. The edge between *who* and *hired* causes this tree to be non-projective. Such a head-modifier relationship is difficult to represent with a CFG, since all words directly or indirectly headed by *hired* (i.e., *who*, *think*, *they*, and *hired*) do not constitute a contiguous sequence of words.

as we will see later in this section.

The most standardly used algorithm for parsing with dependency grammars is presented in (Eisner, 1996; Eisner and Satta, 1999). It runs in time $O(n^3)$, where n is the length of the sentence. Their algorithm exploits the special properties of dependency trees to reduce the worst-case complexity of bilexical parsing, which otherwise requires $O(n^4)$ for bilexical constituency-based parsing. While it seems difficult to improve the asymptotic running time of the Eisner algorithm beyond what is presented in (Eisner and Satta, 1999), McDonald et al. (2005b) show $O(n^2)$-time parsing is possible if trees are not required to be projective. This relaxation entails that dependencies may cross each other rather than being required to be nested, as shown in Fig. 1. More formally, a non-projective tree is any tree that does not satisfy the following definition of a projective tree:

Definition. Let $\mathbf{x} = x_1 \cdots x_n$ be an input sentence, and let $\mathbf{y}$ be a rooted tree represented as a set in which each element $(i, j) \in \mathbf{y}$ is an ordered pair of word indices of $\mathbf{x}$ that defines a dependency relation between a head x_i and a modifier x_j. By definition, the tree $\mathbf{y}$ is said to be projective if each dependency (i, j) satisfies the following property: each word in $x_{i+1} \cdots x_{j-1}$ (if $i < j$) or in $x_{j+1} \cdots x_{i-1}$ (if $j < i$) is a descendent of head word x_i.

This relaxation is key to computational efficiency, since the parser does not need to keep track of whether dependencies assemble into contiguous spans. It is also linguistically desirable in the case of free word order languages such as Czech, Dutch, and German. Non-projective dependency structures are sometimes even needed for languages like English, e.g., in the case of the wh-movement shown in Fig. 1. For languages

with relatively rigid word order such as English, there may be some concern that searching the space of non-projective dependency trees, which is considerably larger than the space of projective dependency trees, would yield poor performance. That is not the case: dependency accuracy for non-projective parsing is 90.2% for English (McDonald et al., 2005b), only 0.7% lower than a projective parser (McDonald et al., 2005a) that uses the same set of features and learning algorithm. In the case of dependency parsing for Czech, (McDonald et al., 2005b) even outperforms projective parsing, and was one of the top systems in the CoNLL-06 shared task in multilingual dependency parsing.

2.1 $O(n^2)$-time dependency parsing for MT

We now formalize weighted non-projective dependency parsing similarly to (McDonald et al., 2005b) and then describe a modified and more efficient version that can be integrated into a phrase-based decoder.

Given the single-head constraint, parsing an input sentence $\mathbf{x} = (x_0, x_1, \cdots, x_n)$ is reduced to labeling each word x_j with an index i identifying its head word x_i. We include the dummy root symbol $x_0 = \langle \text{root} \rangle$ so that each word can be a modifier. We score each dependency relation using a standard linear model

$$s(i,j) = \lambda \cdot \mathbf{f}(i,j) \tag{1}$$

whose weight vector λ is trained using MIRA (Crammer and Singer, 2003) to optimize dependency parsing accuracy (McDonald et al., 2005a). As is commonly the case in statistical parsing, the score of the full tree is decomposed as the sum of the score of all edges:

$$s(\mathbf{x},\mathbf{y}) = \sum_{(i,j)\in\mathbf{y}} \lambda \cdot \mathbf{f}(i,j) \tag{2}$$

When there is no need to ensure projectivity, one can independently select the highest scoring edge (i,j) for each modifier x_j, yet we generally want to ensure that the resulting structure is a tree, i.e., that it does not contain any circular dependencies. This optimization problem is a known instance of the maximum spanning tree (MST) problem. In our case, the graph is directed—indeed, the equality $s(i,j) = s(j,i)$ is generally not true and would be linguistically aberrant—so the problem constitutes an instance of the less-known MST problem for directed graphs. This problem is solved with the Chu-Liu-Edmonds (CLE) algorithm (Chu and Liu, 1965; Edmonds, 1967).

Formally, we represent the graph $G = (V,E)$ with a vertex set $V = \mathbf{x} = \{x_0, \cdots, x_n\}$ and a set of directed edges $E = [0,n] \times [1,n]$, in which each edge (i,j), representing the dependency $x_i \rightarrow x_j$, is assigned a score $s(i,j)$. Finding the spanning tree $\mathbf{y} \subset E$ rooted at x_0 that maximizes $s(\mathbf{x},\mathbf{y})$ as defined in Equation 2 has a straightforward solution in $O(n^2 \log(n))$ time for dense graphs such as G, though Tarjan (1977) shows that the problem can be solved in $O(n^2)$. Hence, non-projective dependency parsing is solved in quadratic time. The main idea behind the CLE algorithm is to first greedily select for each word x_j the incoming edge (i,j) with highest score, then to successively repeat the following two steps: (a) identify a loop in the graph, and if there is none, halt; (b) contract the loop into a single vertex, and update scores for edges coming in and out of the loop. Once all loops have been eliminated, the algorithm maps back the maximum spanning tree of the contracted graph onto the original graph G, and it can be shown that this yields a spanning tree that is optimal with respect to G and s (Georgiadis, 2003).

The greedy approach of selecting the highest scoring edge (i,j) for each modifier x_j can easily be applied left-to-right during phrase-based decoding, which proceeds in the same order. For each hypothesis expansion, our decoder generates the following information for the new hypothesis h:

- a partial translation $\mathbf{x}$;
- a coverage set of input words $\mathbf{c}$;
- a translation score σ.

In the case of non-projective dependency parsing, we need to maintain additional information for each word x_j of the partial translation $\mathbf{x}$:

- a predicted POS tag t_j;
- a dependency score s_j.

Dependency scores s_j are initialized to $-\infty$. Each time a new word is added to a partial hypothesis, the decoder executes the routine shown in Table 1. To avoid cluttering the pseudo-code, we make here the simplifying assumption that each hypothesis expansion adds exactly one word, though the real implementation supports the case of phrases of any length. Line 3 determines whether the translation hypothesis is complete, in which case it explicitly builds the graph G and

Decoding: hypothesis expansion step.

1. Inferer generates new hypothesis $h = (\mathbf{x}, \mathbf{c}, \sigma)$
2. $j \leftarrow |\mathbf{x}| - 1$
3. $t_j \leftarrow \text{tagger}(x_{j-3}, \cdots, x_j)$
4. **if** *complete*$(\mathbf{c})$
5. *Chu-Liu-Edmonds*(h)
6. **else**
7. **for** $i = 1$ **to** j
8. $s_j = \max(s_j, s(i,j))$
9. $s_i = \max(s_i, s(j,i))$

Table 1: Hypothesis expansion with dependency scoring.

finds the maximum spanning tree. Note that it is impractical to identify loops each time a new word is added to a translation hypothesis, since this requires explicitly storing the dense graph G, which would require an $O(n^2)$ copy operation during each hypothesis expansion; this would of course increase time and space complexity (the max operation in lines 8 and 9 only keeps the current best scoring edges). If there is any loop, the dependency score is adjusted in the last hypothesis expansion. In practice, we delay the computation of dependency scores involving word x_j until tag t_{j+1} is generated, since dependency parsing accuracy is particularly low (-0.8%) when the next tag is unknown.

We found that dependency scores with or without loop elimination are generally close and highly correlated, and that MT performance without final loop removal was about the same (generally less than 0.2% BLEU). While it seems that loopy graphs are undesirable when the goal is to obtain a syntactic analysis, that is not necessarily the case when one just needs a language modeling score.

2.2 Features for dependency parsing

In our experiments, we use sets of features that are similar to the ones used in the McDonald parser, though we make a key modification that yields an asymptotic speedup that ensures a genuine $O(n^2)$ running time.

The three feature sets that were used in our experiments are shown in Table 2. We write h-word, h-pos, m-word, m-pos to refer to head and modifier words and POS tags, and append a numerical value to shift the word offset either to the left or to the right (e.g., h-pos+1 is the POS to the right of the head word). We use the symbol $\wedge$ to represent feature conjunctions. Each feature in the table has a distinct identifier, so that, e.g., the POS features

Unigram features:
h-word, h-pos, h-word $\wedge$ h-pos,
m-word, m-pos, m-word $\wedge$ m-pos
Bigram features:
h-word $\wedge$ m-word, h-pos $\wedge$ m-pos,
h-word $\wedge$ h-pos $\wedge$ m-word, h-word $\wedge$ h-pos $\wedge$ m-pos,
m-word $\wedge$ m-pos $\wedge$ h-word, m-word $\wedge$ m-pos $\wedge$ h-pos,
h-word $\wedge$ h-pos $\wedge$ m-word $\wedge$ m-pos
Adjacent POS features:
h-pos $\wedge$ h-pos+1 $\wedge$ m-pos$-$1 $\wedge$ m-pos,
h-pos $\wedge$ h-pos+1 $\wedge$ m-pos $\wedge$ m-pos+1,
h-pos$-$1 $\wedge$ h-pos $\wedge$ m-pos$-$1 $\wedge$ m-pos,
h-pos$-$1 $\wedge$ h-pos $\wedge$ m-pos $\wedge$ m-pos+1
In-between POS features:
if $i < j$:
h-pos $\wedge$ h-pos+k $\wedge$ m-pos $\quad k \in [i, min(i+5, j)]$
h-pos $\wedge$ m-pos$-k$ $\wedge$ m-pos $\quad k \in [max(i, j-5), j]$
if $i > j$:
m-pos $\wedge$ m-pos+k $\wedge$ h-pos $\quad k \in [j, min(j+5, i)]$
m-pos $\wedge$ h-pos$-k$ $\wedge$ h-pos $\quad k \in [max(j, i-5), i]$

Table 2: Features for dependency parsing. It is quite similar to the McDonald (2005a) feature set, except that it does not include the set of all POS tags that appear between each candidate head-modifier pair (i, j). This modification is essential in order to make our parser run in true $O(n^2)$ time, as opposed to (McDonald et al., 2005b).

SOURCE	IDS	GENRE	SENTENCES
English CTB	050–325	newswire	3027
English ATB	all	newswire	13628
OntoNotes	all	broadcast news	14056
WSJ	02–21	financial news	39832
Total			70543

Table 3: Characteristics of our training data. The second column identifies documents and sections selected for training.

h-pos are all distinct from m-pos features.[3]

The primary difference between our feature sets and the ones of McDonald et al. is that their set of "in between POS features" includes the set of all tags appearing between each pair of words. Extracting all these tags takes time $O(n)$ for any arbitrary pair (i, j). Since i and j are both free variables, feature computation in (McDonald et al., 2005b) takes time $O(n^3)$, even though parsing itself takes $O(n^2)$ time. To make our parser genuinely $O(n^2)$, we modified the set of in-between POS features in two ways. First, we restrict extraction of in-between POS tags to those words that appear within a window of five words relative to either the head or the modifier. While this change alone ensures that feature extraction is now $O(1)$ for each word pair, this causes a fairly high drop of performance (dependency accuracy

[3] In addition to these basic features, we follow McDonald in conjoining most features with two extra pieces of information: a boolean variable indicating whether the modifier attaches to the left or to the right, and the binned distance between the two words.

ALGORITHM	TIME	SETUP	TRAINING	TESTING	ACCURACY
Projective	$O(n^3)$	Parsing	WSJ(02-21)	WSJ(23)	90.60
Chu-Liu-Edmonds	$O(n^3)$	Parsing	WSJ(02-21)	WSJ(23)	89.64
Chu-Liu-Edmonds	$O(n^2)$	Parsing	WSJ(02-21)	WSJ(23)	89.32
Local classifier	$O(n^2)$	Parsing	WSJ(02-21)	WSJ(23)	89.15
Projective	$O(n^3)$	MT	CTB(050-325)	CTB(001-049)	86.33
Chu-Liu-Edmonds	$O(n^3)$	MT	CTB(050-325)	CTB(001-049)	85.68
Chu-Liu-Edmonds	$O(n^2)$	MT	CTB(050-325)	CTB(001-049)	85.43
Local classifier	$O(n^2)$	MT	CTB(050-325)	CTB(001-049)	85.22
Projective	$O(n^3)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	87.40(**)
Chu-Liu-Edmonds	$O(n^3)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.79
Chu-Liu-Edmonds	$O(n^2)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.45(*)
Local classifier	$O(n^2)$	MT	CTB(050-325), WSJ(02-21), ATB, OntoNotes	CTB(001-049)	86.29

Table 4: Dependency parsing experiments on test sentences of any length. The projective parsing algorithm is the one implemented as in (McDonald et al., 2005a), which is known as one of the top performing dependency parsers for English. The $O(n^3)$ non-projective parser of (McDonald et al., 2005b) is slightly more accurate than our version, though ours runs in $O(n^2)$ time. "Local classifier" refers to non-projective dependency parsing without removing loops as a post-processing step. The result marked with (*) identifies the parser used for our MT experiments, which is only about 1% less accurate than a state-of-the-art dependency parser (**).

on our test was down 0.9%). To make our genuinely $O(n^2)$ parser almost as accurate as the non-projective parser of McDonald et al., we conjoin each in-between POS with its position relative to (i, j). This relatively simple change reduces the drop in accuracy to only 0.34%.[4]

3 Dependency parsing experiments

In this section, we compare the performance of our parsing model to the ones of McDonald et al. Since our MT test sets include newswire, web, and audio, we trained our parser on different genres. Our training data includes newswire from the English translation treebank (LDC2007T02) and the English-Arabic Treebank (LDC2006T10), which are respectively translations of sections of the Chinese treebank (CTB) and Arabic treebank (ATB). We also trained the parser on the broadcast-news treebank available in the OntoNotes corpus (LDC2008T04), and added sections 02-21 of the WSJ Penn treebank. Documents 001-040 of the English CTB data were set aside to constitute a test set for newswire texts. Our other test set is the standard Section 23 of the Penn treebank. The splits and amounts of data used for training are displayed in Table 3.

Parsing experiments are shown in Table 4. We distinguish two experimental conditions: Parsing and MT. For Parsing, sentences are cased and tokenization abides to the PTB segmentation as used in the Penn treebank version 3. For the MT setting, texts are all lower case, and tokenization was changed to improve machine translation (e.g., most hyphenated words were split). For this setting, we also had to harmonize the four treebanks. The most crucial modification was to add NP internal bracketing to the WSJ (Vadas and Curran, 2007), since the three other treebanks contain that information. Treebanks were also transformed to be consistent with MT tokenization. We evaluate MT parsing models on CTB rather than on WSJ, since CTB contains newswire and is thus more representative of MT evaluation conditions.

To obtain part-of-speech tags, we use a state-of-the-art maximum-entropy (CMM) tagger (Toutanova et al., 2003). In the Parsing setting, we use its best configuration, which reaches a tagging accuracy of 97.25% on standard WSJ test data. In the MT setting, we need to use a less effective tagger, since we cannot afford to perform Viterbi inference as a by-product of phrase-based decoding. Hence, we use a simpler tagging model that assigns tag t_i to word x_i by only using features of words $x_{i-3} \cdots x_i$, and that does not condition any decision based on any preceding or next tags (t_{i-1}, etc.). Its performance is 95.02% on the WSJ, and 95.30% on the English CTB. Additional experiments reveal two main contributing factors to this drop on WSJ: tagging uncased texts reduces tagging accuracy by about 1%, and using only word-based features further reduces it by 0.6%.

Table 4 shows that the accuracy of our truly

[4]We need to mention some practical considerations that make feature computation fast enough for MT. Most features are precomputed before actual decoding. All target-language words to appear during beam search can be determined in advance, and all their unigram feature scores are precomputed. For features conditioned on both head and modifier, scores are cached whenever possible. The only features that are not cached are the ones that include contextual POS tags, since their miss rate is relatively high.

$O(n^2)$ parser is only .25% to .34% worse than the $O(n^3)$ implementation of (McDonald et al., 2005b).[5] Compared to the state-of-the-art projective parser as implemented in (McDonald et al., 2005a), performance is 1.28% lower on WSJ, but only 0.95% when training on all our available data and using the MT setting. Overall, we believe that the drop of performance is a reasonable price to pay considering the computational constraints imposed by integrating the dependency parser into an MT decoder.

The table also shows a gain of more than 1% in dependency accuracy by adding ATB, OntoNotes, and WSJ to the English CTB training set. The four sources were assigned non-uniform weights: we set the weight of the CTB data to be 10 times larger than the other corpora, which seems to work best in our parsing experiments. While this improvement of 1% may seem relatively small considering that the amount of training data is more than 20 times larger in the latter case, it is quite consistent with previous findings in domain adaptation, which is known to be a difficult task. For example, (Daume III, 2007) shows that training a learning algorithm on the weighted union of different data sets (which is basically what we did) performs almost as well as more involved domain adaptation approaches.

4 Machine translation experiments

In our experiments, we use a re-implementation of the Moses phrase-based decoder (Koehn et al., 2007). We use the standard features implemented almost exactly as in Moses: four translation features (phrase-based translation probabilities and lexically-weighted probabilities), word penalty, phrase penalty, linear distortion, and language model score. We also incorporated the lexicalized reordering features of Moses, in order to experiment with a baseline that is stronger than the default Moses configuration.

The language pair for our experiments is Chinese-to-English. The training data consists of about 28 million English words and 23.3 million Chinese words drawn from various news parallel corpora distributed by the Linguistic Data Consortium (LDC). In order to provide experiments comparable to previous work, we used the same corpora as (Wang et al., 2007): LDC2002E18, LDC2003E07, LDC2003E14, LDC2005E83, LDC2005T06, LDC2006E26, LDC2006E8, and LDC2006G05. Chinese words were automatically segmented with a conditional random field (CRF) classifier (Chang et al., 2008) that conforms to the Chinese Treebank (CTB) standard.

In order to train a competitive baseline given our computational resources, we built a large 5-gram language model using the Xinhua and AFP sections of the Gigaword corpus (LDC2007T40) in addition to the target side of the parallel data. This data represents a total of about 700 million words. We manually removed documents of Gigaword that were released during periods that overlap with those of our development and test sets. The language model was smoothed with the modified Kneser-Ney algorithm as implemented in (Stolcke, 2002), and we only kept 4-grams and 5-grams that occurred at least three times in the training data.[6]

For tuning and testing, we use the official NIST MT evaluation data for Chinese from 2002 to 2008 (MT02 to MT08), which all have four English references for each input sentence. We used the 1082 sentences of MT05 for tuning and all other sets for testing. Parameter tuning was done with minimum error rate training (Och, 2003), which was used to maximize BLEU (Papineni et al., 2001). Since MERT is prone to search errors, especially with large numbers of parameters, we ran each tuning experiment three times with different initial conditions. We used n-best lists of size 200 and a beam size of 200. In the final evaluations, we report results using both TER (Snover et al., 2006) and the original BLEU metric as described in (Papineni et al., 2001). All our evaluations are performed on uncased texts.

The results for our translation experiments are shown in Table 5. We compared two systems: one with the set of features described earlier in this section. The second system incorporates one additional feature, which is the dependency language

[5] Note that our results on WSJ are not exactly the same as those reported in (McDonald et al., 2005b), since we used slightly different head finding rules. To extract dependencies from treebanks, we used the LTH Penn Converter (`http://nlp.cs.lth.se/pennconverter/`), which extracts dependencies that are almost identical to those used for the CoNLL-2008 Shared Task. We constrain the converter not to use functional tags found in the treebanks, in order to make it possible to use automatically parsed texts (i.e., perform self-training) in future work.

[6] We found that sections of Gigaword other than Xinhua and AFP provide almost no improvement in our experiments. By leaving aside the other sections, we were able to increase the order of the language model to 5-gram and perform relatively little pruning. This LM required 16GB of RAM during training.

BLEU[%]						
DEP. LM	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
no	33.42	33.38	33.13	36.21	32.16	24.83
yes	34.19 (+.77**)	33.85 (+.47)	33.73 (+.6*)	36.67 (+.46*)	32.84 (+.68**)	24.91 (+.08)
TER[%]						
DEP. LM	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
no	57.41	58.07	57.32	56.09	57.24	61.96
yes	56.27 (−1.14**)	57.15 (−.92**)	56.09 (−1.23**)	55.30 (−.79**)	56.05 (−1.19**)	61.41 (−.55*)
	MT05 (tune)	MT02	MT03	MT04	MT06	MT08
Sentences	1082	878	919	1788	1664	1357

Table 5: MT experiments with and without a dependency language model. We use randomization tests (Riezler and Maxwell, 2005) to determine significance: differences marked with a (*) are significant at the $p \leq .05$ level, and those marked as (**) are significant at the $p \leq .01$ level.

model score computed with the dependency parsing algorithm described in Section 2. We used the dependency model trained on the English CTB and ATB treebank, WSJ, and OntoNotes.

We see that the Moses decoder with integrated dependency language model systematically outperforms the Moses baseline. For BLEU evaluations, differences are significant in four out of six cases, and in the case of TER, all differences are significant. Regarding the small difference in BLEU scores on MT08, we would like to point out that tuning on MT05 and testing on MT08 had a rather adverse effect with respect to translation length: while the two systems are relatively close in terms of BLEU scores (24.83 and 24.91, respectively), the dependency LM provides a much bigger gain when evaluated with BLEU precision (27.73 vs. 28.79), i.e., by ignoring the brevity penalty. On the other hand, the difference on MT08 is significant in terms of TER.

Table 6 provides experimental results on the NIST test data (excluding the tuning set MT05) for each of the three genres: newswire, web data, and speech (broadcast news and conversation). The last column displays results for all test sets combined. Results do not suggest any noticeable difference between genres, and the dependency language model provides significant gains on all genres, despite the fact that this model was primarily trained on news data.

We wish to emphasize that our positive results are particularly noteworthy because they are achieved over a baseline incorporating a competitive 5-gram language model. As is widely acknowledged in the speech community, it can be difficult to outperform high-order n-gram models in large-scale experiments. Finally, we quantified the effective running time of our phrase-based decoder with and without our dependency language

BLEU[%]				
DEP. LM	newswire	web	speech	all
no	32.86	21.75	36.88	32.29
yes	33.19 (+0.33)	22.64 (+0.89)	37.51 (+0.63)	32.74 (+0.45)
TER[%]				
DEP. LM	newswire	web	speech	all
no	57.73	62.64	55.16	58.02
yes	56.73 (−1)	61.97 (−0.67)	54.26 (−0.9)	57.10 (−0.92)
	newswire	web	speech	all
Sentences	4006	1149	1451	6606

Table 6: Test set performances on MT02-MT04 and MT06-MT08, where the data was broken down by genre. Given the large amount of test data involved in this table, all these results are statistically highly significant ($p \leq .01$).

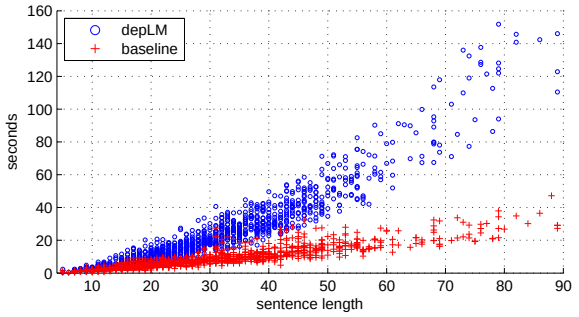

Figure 2: Running time of our phrase-based decoder with and without quadratic-time dependency LM scoring.

model using MT05 (Fig. 2). In both settings, we selected the best tuned model, which yield the performance shown in the first column of Table 5. Our decoder was run on an AMD Opteron Processor 2216 with 16GB of memory, and without resorting to any rescoring method such as cube pruning. In the case of English translations of 40 words and shorter, the baseline system took 6.5 seconds per sentence, whereas the dependency LM system spent 15.6 seconds per sentence, i.e., 2.4 times the baseline running time. In the case of translations

longer than 40 words, average speeds were respectively 17.5 and 59.5 seconds per sentence, i.e., the dependency was only 3.4 times slower.[7]

5 Related work

Perhaps due to the high computational cost of synchronous CFG decoding, there have been various attempts to exploit syntactic knowledge and hierarchical structure in other machine translation experiments that do not require chart parsing. Using a reranking framework, Och et al. (2004) found that various types of syntactic features provided only minor gains in performance, suggesting that phrase-based systems (Och and Ney, 2004) should exploit such information during rather than after decoding. Wang et al. (2007) sidestep the need to operate large-scale word order changes during decoding (and thus lessening the need for syntactic decoding) by rearranging input words in the training data to match the syntactic structure of the target language. Finally, Birch et al. (2007) exploit factored phrase-based translation models to associate each word with a supertag, which contains most of the information needed to build a full parse. When combined with a supertag n-gram language model, it helps enforce grammatical constraints on the target side.

There have been various attempts to reduce the computational expense of syntactic decoding, including multi-pass decoding approaches (Zhang and Gildea, 2008; Petrov et al., 2008) and rescoring approaches (Huang and Chiang, 2007). In the latter paper, Huang and Chiang introduce rescoring methods named "cube pruning" and "cube growing", which first use a baseline decoder (either synchronous CFG or a phrase-based system) and no LM to generate a hypergraph, and then rescoring this hypergraph with a language model. Huang and Chiang show significant speed increases with little impact on translation quality. We believe that their approach is orthogonal (and possibly complementary) to our work, since our paper proposes a new model for fully-integrated decoding that increases MT performance, and does not rely on rescoring.

[7] We note that our Java-based decoder is research rather than industrial-strength code and that it could be substantially optimized. Hence, we think the reader should pay more attention to relative speed differences between the two systems rather than absolute timings.

6 Conclusion and future work

In this paper, we presented a non-projective dependency parser whose time-complexity of $O(n^2)$ improves upon the cubic time implementation of (McDonald et al., 2005b), and does so with little loss in dependency accuracy (.25% to .34%). Since this parser does not need to enforce projectivity constraints, it can easily be integrated into a phrase-based decoder during search (rather than during rescoring). We use dependency scores as an extra feature in our MT experiments, and found that our dependency model provides significant gains over a competitive baseline that incorporates a large 5-gram language model (0.92% TER and 0.45% BLEU absolute improvements).

We plan to pursue other research directions using dependency models discussed in this paper. While we use a dependency language model to exemplify the use of hierarchical structure within phrase based decoders, we could extend this work to incorporate dependency features of both source- and target side. Since parsing of the source is relatively inexpensive compared to the target side, it would be relatively easy to condition head-modifier dependencies not only on the two target words, but also on their corresponding Chinese words and their relative positions in the Chinese tree. This would enable the decoder to capture syntactic reordering without requiring trees to be isomorphic or even projective. It would also be interesting to apply these models to target languages that have free word order, which would presumably benefit more from the flexibility of non-projective dependency models.

Acknowledgements

The authors wish to thank the anonymous reviewers for their helpful comments on an earlier draft of this paper, and Daniel Cer for his implementation of Phrasal, a phrase-based decoder similar to Moses. This paper is based on work funded by the Defense Advanced Research Projects Agency through IBM. The content does not necessarily reflect the views of the U.S. Government, and no official endorsement should be inferred.

References

A. Birch, M. Osborne, and P. Koehn. 2007. CCG supertags in factored statistical machine translation. In *Proc. of the Workshop on Statistical Machine Translation*, pages 9–16.

T. Brants, A. Popat, P. Xu, F. Och, and J. Dean. 2007. Large language models in machine translation. In *Proc. of EMNLP-CoNLL*, pages 858–867.

P. Chang, M. Galley, and C. Manning. 2008. Optimizing Chinese word segmentation for machine translation performance. In *Proc. of the ACL Workshop on Statistical Machine Translation*, pages 224–232.

D. Chiang. 2005. A hierarchical phrase-based model for statistical machine translation. In *Proc. of ACL*, pages 263–270.

Y. J. Chu and T. H. Liu. 1965. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400.

K. Crammer and Y. Singer. 2003. Ultraconservative online algorithms for multiclass problems. *Journal of Machine Learning Research*, 3:951–991.

H. Daume III. 2007. Frustratingly easy domain adaptation. In *Proc. of ACL*, pages 256–263.

Y. Ding and M. Palmer. 2005. Machine translation using probabilistic synchronous dependency insertion grammars. In *Proc. of ACL*, pages 541–548.

J. Edmonds. 1967. Optimum branchings. *Research of the National Bureau of Standards*, 71B:233–240.

J. Eisner and G. Satta. 1999. Efficient parsing for bilexical context-free grammars and head-automaton grammars. In *Proc. of ACL*, pages 457–464.

J. Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*, pages 340–345.

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

L. Georgiadis. 2003. Arborescence optimization problems solvable by Edmonds' algorithm. *Theoretical Computer Science*, 301(1-3):427–437.

L. Huang and D. Chiang. 2007. Forest rescoring: Faster decoding with integrated language models. In *Proc. of ACL*, pages 144–151.

L. Huang, H. Zhang, and D. Gildea. 2005. Machine translation as lexicalized parsing with hooks. In *Proc. of the International Workshop on Parsing Technology*, pages 65–73.

K. Knight. 1999. Decoding complexity in word-replacement translation models. *Computational Linguistics*, 25(4):607–615.

P. Koehn, F. Och, and D. Marcu. 2003. Statistical phrase-based translation. In *Proc. of NAACL*.

P. Koehn, H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, C. Dyer, O. Bojar, A. Constantin, and E. Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proc. of ACL, Demonstration Session*.

D. Marcu, W. Wang, A. Echihabi, and K. Knight. 2006. SPMT: Statistical machine translation with syntactified target language phrases. In *Proc. of EMNLP*, pages 44–52.

R. McDonald, K. Crammer, and F. Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*, pages 91–98.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*, pages 523–530.

J. Nivre. 2003. An efficient algorithm for projective dependency parsing. In *Proc. of the International Workshop on Parsing Technologies (IWPT 03)*, pages 149–160.

F. Och and H. Ney. 2004. The alignment template approach to statistical machine translation. *Computational Linguistics*, 30(4):417–449.

F. Och, D. Gildea, S. Khudanpur, A. Sarkar, K. Yamada, A. Fraser, S. Kumar, L. Shen, D. Smith, K. Eng, V. Jain, Z. Jin, and D. Radev. 2004. A smorgasbord of features for statistical machine translation. In *Proceedings of HLT-NAACL*.

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

K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 2001. BLEU: a method for automatic evaluation of machine translation. In *Proc. of ACL*.

S. Petrov, A. Haghighi, and D. Klein. 2008. Coarse-to-fine syntactic machine translation using language projections. In *Proc. of EMNLP*, pages 108–116.

C. Quirk, A. Menezes, and C. Cherry. 2005. Dependency treelet translation: syntactically informed phrasal SMT. In *Proc. of ACL*, pages 271–279.

A. Ratnaparkhi. 1997. A linear observed time statistical parser based on maximum entropy models. In *Proc. of EMNLP*.

S. Riezler and J. Maxwell. 2005. On some pitfalls in automatic evaluation and significance testing for MT. In *Proc. of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 57–64.

L. Shen, J. Xu, and R. Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proc. of ACL*, pages 577–585.

M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proc. of AMTA*, pages 223–231.

A. Stolcke. 2002. SRILM – an extensible language modeling toolkit. In *Proc. Intl. Conf. on Spoken Language Processing (ICSLP–2002)*.

R. Tarjan. 1977. Finding optimum branchings. *Networks*, 7:25–35.

K. Toutanova, D. Klein, C. Manning, and Y. Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proc. of NAACL*, pages 173–180.

D. Vadas and J. Curran. 2007. Adding noun phrase structure to the Penn treebank. In *Proc. of ACL*, pages 240–247.

C. Wang, M. Collins, and P. Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proc. of EMNLP-CoNLL*, pages 737–745.

D. Wu. 1996. A polynomial-time algorithm for statistical machine translation. In *Proc. of ACL*.

H. Zhang and D. Gildea. 2008. Efficient multi-pass decoding for synchronous context free grammars. In *Proc. of ACL*, pages 209–217.