Syntactic Preprocessing for Statistical Machine Translation

Nizar Habash

Center for Computational Learning Systems
Columbia University
475 Riverside Drive, New York, NY 10115
USA
habash@cs.columbia.edu

Abstract

We describe an approach to automatic source-language syntactic preprocessing in the context of Arabic-English phrase-based machine translation. Source-language labeled dependencies, that are word aligned with target language words in a parallel corpus, are used to automatically extract syntactic reordering rules in the same spirit of Xia and McCord (2004) and Zhang et al. (2007). The extracted rules are used to reorder the source-language side of the training and test data. Our results show that when using monotonic decoding and translations for unigram source-language phrases only, source-language reordering gives very significant gains over no reordering (25% relative increase in BLEU score). With decoder distortion turned on and with access to all phrase translations, the differences in BLEU scores are diminished. However, an analysis of sentence-level BLEU scores shows reordering outperforms no-reordering in over 40% of the sentences. These results suggest that the approach holds big promise but much more work on Arabic parsing may be needed.

1 Introduction

Much research has been done on utilizing syntactic resources within statistical MT (SMT). The work presented here fits within one class of approaches that focus on source-language[1] reordering in the effort to minimize the syntactic differences between source and target languages, and thus learning better translation models (Xia and McCord, 2004; Collins et al., 2005; Zhang et al., 2007; Crego and Mariño, 2007). In our approach, source-language labeled dependencies that are word aligned with target language words in a parallel corpus are used to automatically extract reordering rules. The rules are used to preprocess all of the source sentences in training and decoding in the same spirit of what Xia and McCord (2004) do, although with some important differences that we discuss in the next section.

The language pair we work with is Arabic-English. Arabic is morpho-syntactically quite different from English and its syntactic parsers are not as developed. Our results show that under time/space limitations, where we would use monotonic decoding (i.e., no reordering is allowed by the decoder) and translations for unigram source-language phrases only (about 5% the size of a typical phrase table), source-language reordering gives very significant gains (of around 25% relative increase in BLEU score (Papineni et al., 2002)). With decoder distortion turned on and with access to all phrases, the differences in BLEU scores are diminished between source-language syntactic reordering and no reordering. However, an analysis of sentence-level BLEU scores shows reordering outperforms no-reordering in over 40% of the sentences. These results suggest that the approach holds big promise but much more work on Arabic parsing and rule design may be needed.

In the next section, we discuss and contrast related work. Section 3 presents issues of Arabic syntax in the context of MT and describes the parser we used. Section 4 describes our approach to rule extraction and application. Section 5 presents and discusses the experimental results.

2 Related Research

Much research has been done on utilizing syntactic information within SMT. The approaches used vary in the place of applying syntactic knowledge: for preprocessing, decoding or n-best rescoring and on the source language, target language or both (see (Collins et al., 2005) and (Xia and McCord, 2004) for a general review of different approaches). The approach presented here fits within the class of source-language preprocessing for SMT. It uses full syntactic dependency representations to learn syntactic preprocessing (reordering) rules automatically. The underlying model we use is that of phrase-based SMT (Koehn, 2004). Phrase-based SMT does not utilize any explicit syntactic information. However, it models syntax implicitly in two important ways: (a.) the phrases, which can be as large as 7 or more words, capture syntactic reordering between the source phrase and target phrase; and (b.) the distortion feature in the decoder allows some reordering to be considered by language model ranking. Distortion is typically controlled by a parameter specifying the maximum distance allowed between any two phrases moved to be direct neighbors. Distortion comes at a cost of decreasing the efficiency of the decoder. This is an advantage to

[1] We use the terms *source* and *target* to mean input language to translate from and output language to translate to, respectively (and unlike their usage in descriptions of noisy-channel-model systems).

source preprocessing since monotonic decoding is polynomial (Tillmann et al., 1997). We describe in this paper some results comparing the value of adding reordering with and without distortion and with different phrase sizes.

Collins et al. (2005) describe a technique for preprocessing German to look more like English syntactically. They used six transformations that are applied on German parsed text to reorder it before passing it on to the standard phrase based system. They show a moderate statistically significant improvement. Our work differs from theirs crucially in that our preprocessing rules are learned automatically.

Xia and McCord (2004) describe an approach for translation from French to English, where reordering rules are acquired automatically using source and target parses and word alignment. The reordering rules they use are in a context-free constituency representation with marked heads. The rules are mostly lexicalized. Xia and McCord (2004) use source and target parses to constraint which word alignments are used for rule extraction. Their results show that there is a positive effect to reordering when the decoder is run monotonically (i.e. without additional distortion-based reordering). The value of reordering is diminished if the decoder is run in a non-monotonic way.

Most recently, Zhang et al. (2007) described a similar approach to Xia and McCord (2004)'s. They use chunking (shallow parsing) to learn reordering rules for Chinese-English SMT. They use unlexicalized context-free chunk tags (XPs) and POS tags on the source side only. They use the intersection of forward and backward Giza++ alignments but without motivating their choice empirically. Most interestingly, they allow all possible learned reordering to be used to create a lattice that is input to the decoder. They do not apply the reordering rules to their training data (unlike Xia and McCord (2004)).[2]

The approach presented here is very similar to both (Xia and McCord, 2004) and (Zhang et al., 2007) except for the following important differences: first, we work with Arabic on the source side, a language that is much more different from English than French and is much more morpho-syntactically complex than Chinese. Secondly, we use full parse trees (not just shallow chunks) on the source side only. Thirdly, our rule representation is a context-free dependency, which is much richer than Zhang et al. (2007)'s but effectively the same as that of (Xia and McCord, 2004) except in three specific ways: our rules are all unlexicalized, the relations between children and verbal parents are labeled and childless nodes are marked to add a bit more information of the bigger rule context. Fourthly, like (Xia and McCord, 2004) but unlike (Zhang et al., 2007), we reorder the training data and do not use lattice expansions for the source sentence. Finally, although our results generally agree with the findings from both, they disagree with (Xia and McCord, 2004) on the issue of using distortion in the decoder. Our results show an additional improvement over basic reordering when distortion is turned on. This is perhaps due to quality of the Arabic parser we used which, although state-of-the-art, has a large room for improvement.

[2] In this volume, Crego and Mariño (2007) describe a similar approach that combines reordering and decoding.

3 Arabic Syntactic Parsing

3.1 Arabic Syntactic Issues

Arabic is a morpho-syntactically complex language with many differences from English. We describe here three prominent syntactic features of Arabic that are relevant to Arabic-English translation and that have motivated some of our decisions in this paper.

First, Arabic words are morphologically complex containing clitics whose translations are represented separately in English and sometimes in a different order. For instance, possessive pronominal enclitics are attached to the noun they modify in Arabic but their translation precedes the English translation of the noun: كتابه *kitAbu+hu*[3] 'book+his → his book'. Other clitics include the definite article ال+ *Al+* 'the', the conjunction و+ *w+* 'and' and the preposition ل+ *l+* 'of/for', among others. Separating some of these clitics have been shown to help SMT (Habash and Sadat, 2006). In this paper we do not investigate which clitics to separate, but instead we use the Penn Arabic Treebank (PATB) (Maamouri et al., 2004) tokenization scheme which splits three classes of clitics only. This scheme is most compatible out-of-the-box with statistical parsers trained on the PATB (However, it does not cliticize the definite article as can be seen in Figure 1). We plan to investigate different tokenization schemes for syntactic preprocessing in future work.

Secondly, Arabic verb subjects may be: (a.) pro-dropped (verb conjugated), (b.) pre-verbal, or (c.) post-verbal. The PATB labels the (a.) and (c.) cases as subjects (SBJ) but labels the (b.) case as a topic (TPC) with an empty category inside the verb phrase (VP).[4] From the point of reordering, the case of V-S order is quite interesting in the context of translation to English. See the example in Figure 1 for an illustration of that order. For small noun phrases (NP), phrase-based SMT might be able to handle the reordering in the phrase table if the verb and subject were seen in training. But this becomes much less likely with very long noun phrases that exceed the size of the phrases in a phrase table. The example in Figure 2 illustrates this point. Bolding and italics are used to mark the verb and subordinating conjunction that surround the subject NP (12 words) in Arabic and what

[3] All Arabic transliterations in this paper are provided in the Buckwalter transliteration scheme (Buckwalter, 2004).

[4] The PATB, as well as traditional Arabic grammar consider the Verb-Subject-Object to be the base order; as such, Arabic VPs always have an embedded subject position.

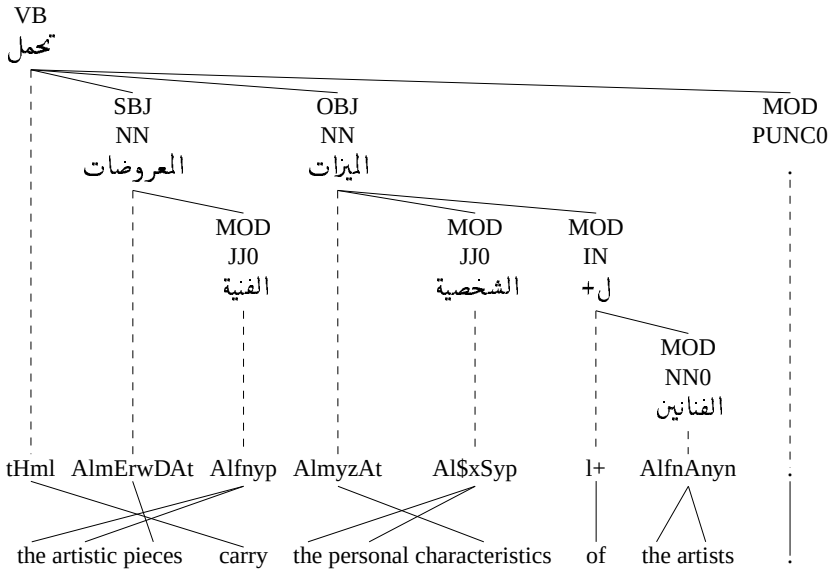

Figure 1: An Arabic dependency tree with the associated alignment to its English translation.

[V اعلن] [NP-SBJ المنسق العام لمشروع السكة الحديد بين دول مجلس التعاون الخليجي حامد خاجه] [SUB ان ...]

[V AEln] [NP-SBJ Almnsq AlEAm lm$rwE Alskp AlHdyd byn dwl mjls AltEAwn Alxlyjy HAmd xAjh] *[SUB An ...]*

[NP-SBJ The general coordinator of the railroad project among the countries of the Gulf Cooperation Council, Hamid Khaja,] **[V announced]** *[SUB that ...]*

Figure 2: An example of long distance reordering of Arabic VSO order to English SVO order

they map to in English, respectively. Additionally, since Arabic is also a pro-drop language, we cannot just move the NP following the verb by default since it can be the object of the verb. We address this issue by using a parser and a labeler to identify the different arguments of a verb.

Finally, Arabic adjectival modifiers typically follow their nouns (with a small exception of some superlative adjectives). However, English adjectival modifiers can follow or precede their nouns depending on the weight of the adjectival phrase: single word adjectives precede but multi-word adjectives phrases follow (or precede while hyphenated). For example, رجل طويل *[NP rajul [AdjP Tawiyl]]* translates as 'a tall man' but رجل طويل القامة *[NP rajul [AdjP Tawiyl AlqAmp]]* translates as 'a man tall of stature'. To address this issue, we introduce POS tag *weights* which distinguish childless words from words with children. See the third, fifth, seventh and last words in the Arabic example in Figure 1. They are all marked as having 0 children.

3.2 Arabic Parsing

A limited amount of research has been done on Arabic parsing compared to English (Bikel, 2002; Kulick et al., 2006). And Arabic parsing performance can still be improved. Parsing research is decidedly outside the scope of this paper; however, as consumers of parsing technology, some of our design decisions were shaped by its current limitations. In this section we describe the different steps and decisions made for producing the parses we use.

3.2.1 Tokenization and POS Tagging

For tokenization, we use the PATB tokenization scheme: 4-way normalized segments into conjunction, particle, word and pronominal clitic. For POS tagging, we use the collapsed tagset for PATB (24 tags), which is what the Bikel parser uses for Arabic. Tokenization and POS tagging are done using the publicly available Morphological Analysis and Disambiguation (MADA) tool (Habash and Rambow, 2005) together with TOKAN, a general tokenizer for Arabic (Habash and Sadat, 2006). MADA tags Arabic words using fourteen sub-tags (a very large tagset – in practice over 2,200 tags). This tagset is first reduced for the parser as described above. Moreover, for rule-extraction and application, we use a further-reduced tagset of 14 tags, that abstracts away all inflectional features (such as verbal tense or nominal number). What optimal tokenization and tagset to use for Arabic-English syntactic reordering is an empirical question that we do not attempt to answer here. We leave it to future work.

3.2.2 Constituency Parsing

For parsing, we use the Bikel parser (Bikel, 2002) trained on the PATB (Part 1).[5] Arabic sentence length, which averages over 35 words with PATB tokenization (in the news genre), slows down the parser and increases its chances of producing null parses.[6] Due to time and resource limitations, we followed the parsing solution proposed in (Habash et al., 2006), where the source POS tagged text is heuristically chunked and the parser is applied only to unique chunks. This solution cuts the parsed text length by over three-fourths (to an average of 8 words) and the processing time by 90%. We diverge from Habash et al. (2006) in using a second set of heuristics to put the chunks back together instead of treating them as sisters in a forest. The heuristics used in dechunking include attaching open and close parentheses to intermediate chunks or linking sentence-final punctuation to the first head in the tree.[7] The default linking is right-branching (the head of chunk n is a child of the head of chunk $n-1$). In PATB-1, right-branching is 6.4 times as common as left branching.

3.2.3 Labeled Weighted Dependencies

The default output of the parser is an unlabeled constituency representation. We convert the constituency representation into a dependency tree by identifying heads using simple head-percolation rules. Dependency parsing accuracy starting with untagged text is 76.2% Chunking and dechunking potentially reduce the accuracy by about 19.2% relative (computed on gold trees). An important advantage of dependencies (as opposed to constituencies) is that they abstract away from phrase-structure possible multiple projections and focus on the headedness relation between words in a sentence. Of course, since we use ordered dependency, there is arguably little difference beyond syntactic sugar between our basic representation and that of (Xia and McCord, 2004). We diverge from (Xia and McCord, 2004) by using relational labels for verb children and by marking childless nodes (we call this *weighing*). Also, unlike Habash et al. (2006), whose dependencies are in a deeper representation that abstracts off syntactic phenomena such as passivization and introduces empty categories, our dependency representation is closer to the surface. We also diverge from Habash et al. (2006) in learning how to assign dependency relation labels automatically rather than using heuristics.

Both weighing and labeling are inspired by the examples we noted earlier on Arabic syntax. Weighing is done deterministically by concatenating a '0' to the end of the POS tag of a word that has no children. As for labeling, we only marked the relations among verbs and their children. For each child of a verb in a dependency tree, we classify its relationship to its parent as one of the following: SBJ, TPC, OBJ and MOD. This label set is a simplified version of the PATB's dashtag set (which includes more information such as LOC and TMP). We automatically learn how to label using the PATB as training data. We use Yamcha (Kudo and Matsumoto, 2003), an implementation of support vector machines which includes dynamic features.[8] For machine learning features, we used the word, its POS tag, its parent, and relative distance from the parent (computed as word location subtracted from parent location). We also use the same features of all the words in a window of +/- 3 words around the focus word. We used the previous three predicted classifications as dynamic features. The automatic verb-child labeler has 94.6% accuracy.

In the rest of this paper, we refer to four kinds of dependencies that are in slight variation: DEP is our basic POS tagged structural dependency without labels or weights; LDEP is a labeled version of DEP; WDEP is a weighted version of DEP; and WLDEP is a weighted labeled dependency. We use WLDEP in all the reported results in this paper, but we contrast the behavior of that representation with its more impoverished variants.

Although we took several consideration of Arabic syntax in the context of English translation when making decisions on what the dependency and its labels should look like, we believe we only scratched the surface of the most common cases. Much more future work will be dedicated to selecting the optimal set of dependency structures and labels.[9]

4 Syntactic Reordering Rules

In this section we describe the process for extracting and applying syntactic reordering rules.

4.1 Extraction of Reordering Rules

Given an aligned sentence pair S and T, and a parse of S (P_S), we compute for each node N in P_S a link range L, which is basically the union of the T word positions to which N and all of its children are aligned. We then traverse the parse tree P_S and extract, at each node with children, a pairing of condition (C) and reordering (R): C is simply the ordered dependency and R is the target order of its nodes. In the case that a dependency node is not linked (typically, a failure in the alignment step), we ignore that child completely. In the case when multiple ordering are possible due to overlapping alignments, we produce all possible reorderings and give them

[5] We did not use a combination of all three parts of the PATB because of known incompatibilities. We used PATB-1 because some of the resources we had were designed for it. These resources needed additional work to handle PATB-2 and PATB-3. We plan to address this issue in future work.

[6] We computed based on a sample that it would take 30 days on a 2GHz Opteron Processor to parse all of our training data.

[7] This code is available to interested parties for research purposes. Please contact the author.

[8] We use Yamcha's default settings: standard SVM with 2nd degree polynomial kernel and 1 slack variable.

[9] Habash et al. (2007) attempt to do this for the task of automatic case prediction in Arabic.

Table 1: Examples of extracted rules

Condition (C)		**Reordering (R)**	$P(R\|C)$
$[VB]_1$,SBJ/NN_2,OBJ/NN_3,MOD/$PUNC_4$	$\Rightarrow$	2,1,3,4	0.55
	$\Rightarrow$	1,2,3,4	0.25
	$\Rightarrow$	2,3,1,4	0.10
$[NN]_1$,MOD/$JJ0_2$	$\Rightarrow$	2,1	0.92
	$\Rightarrow$	1,2	0.08
$[NN]_1$,MOD/$JJ0_2$,MOD/IN_3	$\Rightarrow$	2,1,3	0.82
	$\Rightarrow$	1,2,3	0.07
	$\Rightarrow$	3,2,1	0.07
$[IN]_1$,MOD/NN_2	$\Rightarrow$	1,2	0.97
	$\Rightarrow$	2,1	0.03

equal probability. The intuition here is that different specific occurrences of C will have different kinds of alignment errors associated with them, but that good reorderings will have high co-occurrence rate and rise to the top claiming more of the probability.

For example, the first node in the parse in Figure 1 is a verb with three children: a subject, an object and a punctuation modifier. The verb word تحمل *tHml* is aligned to the fourth word on the English side 'carry'. The subject noun phrase, headed by the noun المعروضات *AlmErwDAt*, is aligned to the first three words. The object noun phrase, headed by the noun الميزات *AlmyzAt*, is aligned to words five to ten on the English side. The Arabic final punctuation is aligned to the English final punctuation. The alignment information tells us that to obtain the English order from the Arabic order, we need to place the verb after the subject. This rule instance learned is presented in the first row in Table 1. The first column specifies the condition. The comma-separated nodes are represented in order using their POS and label. The head is unlabeled and is marked with square brackets. The nodes in the condition are unlexicalized (although this approach can be easily extended to handle lexicalized rules). The subscript numerals are unique identifiers that allow us to link the nodes in the condition to the reordering position. These identifiers are basically the ordered position of the node in the condition. The second column is the associated reordering presented using the unique node identifiers. The last column presents the conditional probability mass of the reordering given the condition (due to space limitations, we only show the top three values). These probabilities are computed over the whole corpus.

4.2 Properties of Reordering Rules

The approach described above for extracting reordering rules is independent of the kind of alignment and the kind of parse representation. We extracted rule sets using the four types of parses we described earlier (DEP, LDEP, WDEP and WLDEP) and using five alignment strategies: Giza's forward (DIR) and backward (INV) alignments, their intersection (INTERX) and union (UNION), and the grow-diag-final (GDF) heuristic used by default in Pharaoh (Koehn, 2004) for phrase extraction. Due to space limitations, we only present two sets of comparisons in which we vary either the dependency type (Table 2) or the alignment (Table 3). In both tables, we list, starting with the second column, the number of extracted rules, the number of unique conditions, the average ambiguity of rules (number of reorderings per condition), the frequency-weighted probability mass present in the set of top-1 reorderings for all conditions, and the coverage rate of the rule set. Coverage is computed by applying the rules to the test data and counting the number of no matches. As mentioned earlier, not all sub-trees are used fully for rule extraction since the lack of alignment is dealt with by dropping the unaligned word. The back-off strategy we describe in the next section resolves the problem of coverage almost completely. So, coverage is not a major consideration for us (plus the variation among the different rule sets is rather small).

Table 2: Rule Properties (with INTERX Alignment)

Parse Type	**Rule Count**	**Unique Cond's**	**Rule Ambig**	**Top-1 Mass**	**Cover**
DEP	53K	31K	1.7	81.6%	95.3%
LDEP	58K	36K	1.6	82.1%	94.7%
WDEP	66K	42K	1.6	82.4%	93.6%
WLDEP	71K	47K	1.5	82.9%	93.0%

From the data presented in both tables, we motivate using the rule set derived using the intersection alignment (similar to (Zhang et al., 2007)) and the weighted labeled dependency. WLDEP combines the lowest rule ambiguity with the best distribution of probability mass focused on the top-1 orderings. The intersection alignment minimizes rule ambiguity because of its high alignment precision, which is more desirable than a high alignment recall (which could lead to many ambiguous reorderings).

Table 3: Rule Properties (with WLDEP Parses)

Align Type	Rule Count	Unique Cond's	Rule Ambig	Top-1 Mass	Cover
INTERX	71K	47K	1.5	82.9%	93.0%
DIR	179K	96K	1.9	70.9%	95.1%
INV	220K	68K	3.2	67.1%	94.1%
GDF	321K	97K	3.3	61.0%	95.2%
UNION	529K	97K	5.5	52.8%	95.2%

4.3 Application of Reordering Rules

When applying the rules, we select the ordering with the highest conditional probability. It is possible to extend this approach to create a weighted lattice of multiple orders that can be passed on to the decoder in a manner similar to Zhang et al. (2007) and Crego and Mariño (2007). We leave this solution to future work.

Starting from the top of the parse tree and descending recursively, the new order of every node and its children is determined by matching features of the node and children to the condition of an existing reordering rule. Since this process is deterministic given the rules, does not allow partial matching, and respects the projectivity of the parse tree, there is no issue of rule order of application.

As far as the issue of coverage discussed earlier, we utilize a stepped back-off mechanism in which, when a condition is not matched, we attempt to match it with a different known condition: we first ignore the POS weight distinction and try to match. If no matching happens, we drop the POS tag altogether. The last thing we ignore is the syntactic label. In the case of complete mismatch, we use Arabic word order. The successful application of back-off brings the coverage to almost 100% to all rule sets examined. This back-off approach could perhaps be improved by allowing partial matches.

5 Results

5.1 Experimental Data and Metrics

All of the training data used here is available from the Linguistic Data Consortium (LDC). We use an Arabic-English parallel corpus[10] consisting of 131K sentence pairs, with approximately 4.1M Arabic tokens and 4.4M English tokens. Word alignment is done with GIZA++ (Och and Ney, 2003). All evaluated systems use the same surface trigram language model, trained on approximately 340 million words of English newswire text from the English Gigaword corpus.[11] English LM preprocessing simply included down-casing, separating punctuation from words and splitting off "'s". Trigram language models are implemented using the SRILM toolkit (Stolcke, 2002).

We use the standard NIST MTEval datasets for the years 2003, 2004 and 2005 (henceforth MT03, MT04 and MT05, respectively).[12] Both BLEU (Papineni et al., 2002) and NIST (Doddington, 2002) metric scores are reported. All scores are computed against four references with n-grams of maximum length four. The first 200 sentences in the 2002 MTEval test set were used for Minimum Error Training (MERT) (Och, 2003).

We contrast three conditions yielding eight different systems: (a.) applying syntactic reordering to both training and testing data (REORDER) or no reordering (BASIC); (b.) using only phrases that have single words on the source side (UNI) or all phrases up to length 7 (ALL); (c.) using monotone decoding (MONO) or allowing distortion in the decoder (with distortion limit = 4) (DIST). The size of the phrase tables in the UNI mode is around 5% the size of the ALL table. MONO decoding is known to be much more efficient that DIST decoding (Tillmann et al., 1997). The same set of rules (INTERX-WLDEP) is used for all REORDER systems. We use Pharaoh in all our experiments (Koehn, 2004). MERT tuning is done separately for each combination. The results are presented in Table 4.

5.2 Discussion

The three varied parameters all address syntax in different ways. The phrase-table parameter (UNI vs. ALL) captures syntactic information implicit in the phrase-table: UNI has little or no syntactic information as opposed to ALL which has plenty lexicalized syntactic information. The decoding parameter (MONO vs. DIST) also presents a model of syntax: MONO does not allow any reordering, whereas DIST allows different choices to be presented to the target language model. Finally, our source-language reordering is a linguistically based automatically learned model for reordering which is contrasted against its own absence in BASIC.

Under the UNI+MONO conditions, REORDER outperforms BASIC by a staggering 25% relative improvement in BLEU score. As either DIST or ALL are included, the difference diminishes completely, with the ALL phrase-table in MONO mode outperforming UNI in DIST mode for either BASIC or REORDER conditions. Finally, comparing BASIC and REORDER with ALL phrases and DIST decoding shows little difference in performance. Except for the UNI+MONO case, all differences in pairings of BASIC and REORDER are not statistically significant. The differences between pairs varying in UNI/ALL or MONO/DIST are all statistically significant. This result suggests that under time/space limitations, where we would use monotonic decoding (i.e., no reordering is allowed by the decoder) and only phrases for unigram source-language phrases (about 5% the size of a typical phrase table), source language re-

[10] The parallel text includes Arabic News (LDC2004T17), eTIRR (LDC2004E72), English translation of Arabic Treebank (LDC2005E46), and Ummah (LDC2004T18).

[11] Distributed by the Linguistic Data Consortium: http://www.ldc.upenn.edu

[12] http://www.nist.gov/speech/tests/mt/

Table 4: Results

	TEST SET					
	MT03		MT04		MT05	
SYSTEM	**BLEU**	**NIST**	**BLEU**	**NIST**	**BLEU**	**NIST**
UNI+MONO+BASIC	26.18	8.02	26.34	8.26	26.12	8.01
UNI+MONO+REORDER	**32.68**	**8.73**	**32.64**	**9.06**	**33.02**	**8.85**
UNI+DIST+BASIC	37.49	9.12	35.12	9.27	37.10	9.22
UNI+DIST+REORDER	37.26	9.17	35.83	9.34	37.29	9.26
ALL+MONO+BASIC	42.80	10.00	39.62	9.92	41.35	10.00
ALL+MONO+REORDER	42.27	9.92	39.79	10.06	41.00	10.01
ALL+DIST+BASIC	46.25	10.42	41.53	10.03	44.66	10.35
ALL+DIST+REORDER	44.36	10.12	41.10	10.06	43.61	10.25
Oracle Combination: ALL+DIST+*	**49.12**	**10.64**	**44.94**	**10.46**	**47.64**	**10.69**
BASIC > REORDER	52.19%		45.68%		49.24%	
BASIC < REORDER	40.42%		48.34%		45.74%	
BASIC = REORDER	7.39%		5.99%		5.02%	

ordering gives very significant gains. However, under other conditions, the differences are not big given current parsing and alignment technology.

In the last row in Table 4, we present the results of an oracle combination that selects the output from either ALL+DIST+BASIC or ALL+DIST+REORDER based on the sentence level BLEU score. The BLEU score of the oracle combination is between 6% and 8% higher than the best of the combined systems. The percentage of sentences in the combination from the BASIC or REORDER variants is quite similar. This indicates that the differences in performance between the two variants are complementary. This also confirms that syntactic preprocessing (even with current parsing performance) improves over basic SMT in over than 40% of the sentences in the test sets.

6 Conclusion and Future Work

We presented an approach to source-language syntactic preprocessing in the context of Arabic-English phrase-based machine translation. Our results suggest that the approach holds big promise but much more work on parsing and rule design may be needed. The results also suggest that this approach can provide a good alternative under speed/space limitations given the presence of a fast efficient parser — a clear trade-off.

In the future, we plan to use better parsers and investigate the optimal parse representation for rule learning in terms of tokenization, dependency structure, POS tags and dependency labels. We also plan to use better alignment techniques that may be useful for rule learning although they have not shown promise in standard phrase-based MT, e.g., (Elming and Habash, 2007). We also plan to investigate different back-off techniques including allowing partial matching. Finally, the results from the oracle combination analysis suggest a direction of using parsing quality information to help with actual combination of systems with and without syntactic preprocessing.

Acknowledgements

This work was funded by DARPA Contract No. HR0011-06-C-0023. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of DARPA. I thank Owen Rambow, Roland Kuhn and George Foster for helpful discussions.

References

Daniel Bikel. 2002. Design of a Multi-lingual, Parallel-processing Statistical Parsing Engine. In *Proceedings of International Conference on Human Language Technology Research (HLT)*, pages 24–27.

Tim Buckwalter. 2004. Buckwalter Arabic Morphological Analyzer Version 2.0. Linguistic Data Consortium, University of Pennsylvania, 2002. LDC Cat alog No.: LDC2004L02, ISBN 1-58563-324-0.

Michael Collins, Philipp Koehn, and Ivona Kucerova. 2005. Clause Restructuring for Statistical Machine Translation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 531–540, Ann Arbor, Michigan.

Josep M. Crego and Jose B. Mariño. 2007. Syntax-enhanced N-gram-based SMT. In *Proceedings of the Machine Translation Summit (MT SUMMIT XI)*, Copenhagen, Denmark.

George Doddington. 2002. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceeding of the ARPA Workshop on Human Language Technolog*.

Jakob Elming and Nizar Habash. 2007. Combination of statistical word alignments based on multiple pre-

processing schemes. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 25–28, Rochester, New York, April. Association for Computational Linguistics.

Nizar Habash and Owen Rambow. 2005. Arabic Tokenization, Part-of-Speech Tagging and Morphological Disambiguation in One Fell Swoop. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 573–580, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Nizar Habash and Fatiha Sadat. 2006. Arabic Preprocessing Schemes for Statistical Machine Translation. In *Proceedings of the 7th Meeting of the North American Chapter of the Association for Computational Linguistics/Human Language Technologies Conference (HLT-NAACL06)*, pages 49–52, New York, NY.

Nizar Habash, Bonnie Dorr, and Christof Monz. 2006. Challenges in Building an Arabic-English GHMT System with SMT Components. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA06)*, pages 56–65, Cambridge, MA.

Nizar Habash, Ryan Gabbard, Owen Rambow, Seth Kulick, and Mitch Marcus. 2007. Determining case in Arabic: Learning complex linguistic behavior requires complex linguistic features. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 1084–1092.

Philipp Koehn. 2004. Pharaoh: a Beam Search Decoder for Phrase-based Statistical Machine Translation Models. In *Proceedings of the Association for Machine Translation in the Americas*, pages 115–124.

Taku Kudo and Yuji Matsumoto. 2003. Fast methods for kernel-based text analysis. In *Proceedings of the 41st Meeting of the Association for Computational Linguistics (ACL'03)*, pages 24–31, Sapporo, Japan.

Seth Kulick, Ryan Gabbard, and Mitch Marcus. 2006. Parsing the arabic treebank: Analysis and improvements. In *Proceedings of the Treebanks and Linguistic Theories Conference*, pages 31–42, Prague, Czech Republic.

Mohamed Maamouri, Ann Bies, and Tim Buckwalter. 2004. The Penn Arabic Treebank: Building a large-scale annotated Arabic corpus. In *NEMLAR Conference on Arabic Language Resources and Tools*, Cairo, Egypt.

Franz Josef Och and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29(1):19–52.

Franz Josef Och. 2003. Minimum Error Rate Training for Statistical Machine Translation. In *Proceedings of the 41st Annual Conference of the Association for Computational Linguistics*, pages 160–167, Sapporo, Japan, July.

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 of the Association for Computational Linguistics*, pages 311–318, Philadelphia, PA.

Andreas Stolcke. 2002. SRILM - an Extensible Language Modeling Toolkit. In *Proceedings of the International Conference on Spoken Language Processing (ICSLP)*, pages 901–904.

Christoph Tillmann, Stephen Vogel, Hermann Ney, and Alex Zubiaga. 1997. A DP-based search using monotone alignments in statistical translation. In *Proceedings of the Thirty-Fifth Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics*, pages 289–296, Madrid, Spain.

Fei Xia and Michael McCord. 2004. Improving a statistical mt system with automatically learned rewrite patterns. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 508–514, Geneva, Switzerland.

Y. Zhang, R. Zens, and H. Ney. 2007. Chunk-level reordering of source language sentences with automatically learned rules for statistical machine translation. In *Proceedings of the Workshop on Syntax and Structure in Statistical Translation at the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1–8, Rochester, NY.