If I Only Had a Parser: Poor Man's Syntax for Hierarchical Machine Translation

David Vilar, Daniel Stein, Stephan Peitz and Hermann Ney

Lehrstuhl für Informatik 6
RWTH Aachen University
Aachen, Germany
{vilar,stein,peitz,ney}@informatik.rwth-aachen.de

Abstract

In the last few years, several enhancements for the hierarchical phrase-based translation model have been proposed. They aim to include additional syntactic information in the translation process in order to achieve better fluency in the generated output.

In this work we review and compare three such methods: parsematch, soft syntactic labels and string-to-dependency. Our goal is to find out if these models complement each other of if they rather address the same deficiencies in the translation process. Furthermore, we present a novel method for extending the translation model in the same direction without the need for parse trees, since they may not be available for some languages. Our approach is based only on automatic clustering of phrases, without the need for additional information. Our findings show that we are able to achieve similar results as when applying syntax models.

1. Introduction

The hierarchical phrase-based model was introduced by [1] as a generalization of the well-known phrase-based translation approach. The hierarchical model is formally defined as a synchronous context free grammar, a property which makes it attractive for possible enhancements aimed at including additional syntactic information in the translation approach. Many different enhancements to the basic model have been proposed in recent years.

In this work we will review and compare three such approaches: parsematch features [2], which mark if the hierarchical phrases match syntactic constructions; a string-to-dependency model [3], which augments the hierarchical rules with dependency information; and soft syntactic labels [4], which enhances the set of non-terminals of the hierarchical models. Our goal will be to determine if these models are complementary or if they address the same issues with the baseline translation model.

Furthermore we present a novel approach which aims to obtain similar improvements as the syntax-based models, but without the need for additional information in the form of parse trees. This may pose an important advantage for certain tasks, for example for translation pairs involving under-resourced languages, for which no linguistic tools may be available. We will compare this new model with the previously listed syntax-based approaches and show that its performance is only slightly worse.

This paper is organized as follows: In Section 2 we shortly review the related work and in Section 3 the hierarchical phrase-based translation approach. In Section 4 we present the three syntactic models we will analyze. Section 5 introduces the new model presented in this paper. The performance of all the approaches is analyzed in Section 6 and conclusions are drawn in Section 7.

2. Related Work

One of the first works to incorporate syntactic knowledge in a statistical machine translation model was [5], although the performance was not on-par with other state-of-the-art approaches at that time. Further development in this direction achieved competitive results, as can be seen in [6] and later publications by the same group.

In contrast to these works, which propose new models centered around the syntactic information, we focus mainly on methods that can be easily incorperated into an existing hierarchical system. Apart from the models employed in this work, other approaches in this field like [7] and [8] are working in similar directions, but create a rather large quantity of features. For these approaches it is not really clear which features are actually beneficial for the translation process.

3. Hierarchical Phrase-based Translation

The hierarchical phrase-based translation approach [1, 9] is a generalization of the well-known phrase-based translation approach [10, 11], where the phrases are allowed to have gaps. The model is formalized as a synchronous context free grammar, with rules of the form

$$X \rightarrow \langle \alpha, \gamma, \sim \rangle \tag{1}$$

where X is a non-terminal symbol, α and γ strings of terminals and non-terminal symbols in the source and target language respectively and $\sim$ is a one-to-one correspondence between the non-terminals of α and γ. An example for a

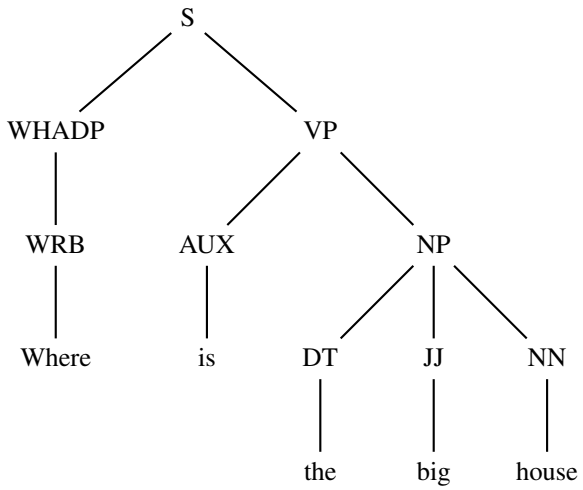

Figure 1: Example of a parse tree.

Chinese-to-English translation task would be:

$$X \rightarrow \langle \text{不是 } X^{\sim 0} \text{ 的 } X^{\sim 1}, \text{ is not the } X^{\sim 1} \text{ of the } X^{\sim 0} \rangle .$$

If α and γ consist only of terminal symbols, we will denote them as *lexical phrases*.

The extraction process consists of two steps. In the first one, we extract the set of lexical phrases. In the second step, we look for those phrases that contain other smaller subphrases. These smaller ones are suppressed and gaps are produced on the larger ones, which correspond to the nonterminals in the grammar.

The translation probabilities, as is standard practice in statistical machine translation nowadays, are modelled using a log-linear approach [12], the scaling factors of which are usually computed by numerically optimizing over a performance measure [13].

4. Syntax-based Models

In this section we will present the three syntax-based models that we will compare in our study. Two of the approaches analyzed apply the concept of *valid syntactic phrases*. Given a monolingual sentence (be it in the source or the target language) and the associated parse tree, we will say that a lexical phrase extracted from this sentence is *syntactically valid* if it corresponds to the yield of one of the nodes in the syntax tree. We extend this concept to hierarchical phrases by defining it as valid if the originating phrase was syntactically valid and every phrase which was suppressed in order to generate the gaps in the phrase is also syntactically valid.

Figure 1 shows an example of a sentence together with the associated syntax tree. The phrase "the big house", for example, is syntactically valid, as it is the yield of the node labelled with NP. Examples of syntactically invalid phrases are "big house" or "is the". "Where $X^{\sim 0}$ the big house" and "Where is $X^{\sim 0}$" are valid hierarchical phrases, whereas "Where $X^{\sim 0}$ big house" or "Where is the $X^{\sim 0}$" are not.

For the syntactically invalid phrases, we can search for the node whose yield is closest to the phrase we are considering. We choose the node for which a minimum number of words has to be added or deleted from the phrase so that it fits the yield of the node. The node is then called the *best match node* for the phrase. In case of ambiguity we favour addition over deletion of words. Returning to the example of Figure 1, the phrase "big house" has NP as best match node, because adding just one word, "the", we arrive at the yield of this node.

4.1. Parsematch Features

In [2] the authors propose to compute additional features which measure how well the extracted phrases correspond to linguistic structures. In contrast to other approaches in which rules are extracted to enforce the syntactical integrity of the translation (e.g. [14]), they do not limit the extraction algorithm. The rule extraction is the same as for the standard hierarchical phrase-based model, but additional scores are computed for the generated phrases. They point out that non-syntactical phrases are necessary to achieve good translation performance, as shown in [11, 6]. It is also worth noting that by adjusting the corresponding scaling factor the minimum error rate training procedure can fall back to the original system.

In contrast to other approaches, which normally only take into account the target syntax, both the source and the target part of the rules can be considered. The inclusion of this information as additional scores in the phrases does hardly have an impact on computation time.

The simplest way to include this information is to add a new binary feature, which is fired if the phrase is syntactically valid, else it has a value of 0. Other features that try to take into account how many words have to be added or removed from a phrase to be syntactically valid have also been investigated in [2]. In this work we further apply the "relative" distance measure, as defined in that paper.

4.2. String-to-Dependency

The second possibility for introducing syntactical information in the translation process is inspired by [3]. The authors propose to augment the phrases used in the translation including dependency information of the target side. At generation time they build a dependency tree of the produced translation and score them using appropriate language models. These dependency language models are able to span longer distances than the standard n-gram language models at the word level.

Figure 2 shows an example dependency tree. A language model that scores this structure can for example evaluate the left-handed dependency of the structure "In", followed by

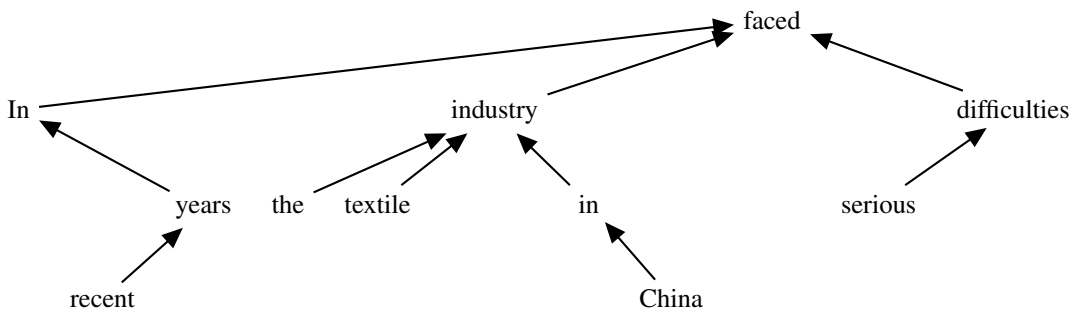

Figure 2: Dependency parsing for the sentence "In recent years, the textile industry in China faced serious difficulties".

"industry", on the structure "faced".

In [3] the authors propose to parse the training data and use only phrases that meet certain restrictions. The first possibility is what the authors call a *fixed* dependency structure. With the exception of one word within this phrase, called the *head*, no outside word may have a dependency within this phrase. Also, all inner words may only depend on each other or on the head. The second possibility is what the authors name a *floating* dependency structure, in which the head dependency word may also exist outside the phrase.

Formally, for the word at position k let dep_k be the position of the word it depends on. A sequence of words spanning from i to j constitutes a *fixed* structure, if there exists $h \in [i, j]$ such that

- $dep_h \notin [i, j]$
- $\forall k \in [i, j] \wedge k \neq h, dep_k \in [i, j]$
- $\forall k \notin [i, j], dep_k = h \vee dep_k \notin [i, j]$

and *floating* with children C for a non-empty set $C \subseteq \{i, \ldots, j\}$ if

- $\exists h \notin [i, j]$, s.t. $\forall k \in C, dep_k = h$
- $\forall k \in [i, j] \wedge k \notin C, dep_k \in [i, j]$
- $\forall k \notin [i, j], dep_k \notin [i, j]$.

The phrases that do not belong to one of these categories are considered invalid and removed from the translation table.

The original authors score the resulting dependency tree already at translation time using dependency language models. In this work we used an alternative approach, but similar in the basic ideas.

We do not impose the above restrictions to the phrase table, rather we mark those phrases that would be rejected with a binary feature. In this way, it is again the minimum error rate training algorithm the one responsible for deciding if these phrases are useful for the translation or not. For each phrase, we store the corresponding dependency information, and further memorize for all hierarchical phrases if the gaps were dependent on the left or on the right side. We then penalize if the new phrase filling the gap in a larger hierarchical rule is pointing into the wrong direction, i.e. is pointing to the left when the hierarchical rule is expecting the gap to point to the right and vice versa. This introduces three features in the log-linear model: one for merging errors to the left, one for merging errors to the right and one for the number of non-valid dependency structures used.

We then generate an n-best list of translations, constructing the dependency tree using the information contained in the augmented phrase set, and score them using three dependency language models: one for left-side dependencies, one for right-side dependencies and one for head structures.

Note that we do *not* parse the produced translation with a standard dependency parser. A standard dependency parser will try to find a "sensible" dependency structure, even for ill-formed sentences. In this way, the language models will not have enough discriminative power to favour the more correct translations. The translation system has additional information at decoding time, and can build dependency trees which are ill-formed for ill-formed sentences. In this way the rescoring process can choose those sentence which have sensible dependency structures attached to them.

One drawback of this approach is that badly reconstructed dependency trees have fewer probabilities to compute. Thus they tend to score higher than better structured trees in other sentences. We saw this effect in early experiments, even if we penalized every reconstruction error. Therefore, we include a language count feature that is incremented each time we compute a dependency language model score. This is similar in spirit as the word penalty.

4.3. Soft Syntactic Labels

Another possibility to include syntax information in the hierarchical model is to extend the set of non-terminals in the hierarchical model from the original set of generic symbols to a richer, syntax-oriented set [15]. With this, we hope to improve the syntactic structure of the output sentence. For example there may be rules which ensure that there is a verb in the translation of every source verb phrase.

However, augmenting the set of non-terminals also restricts the parsing space and thus we alter the set of possible translations. Furthermore, it can happen that no parse can be found for some input sentences. To address this issue, our extraction is extended in a similar way as in the work of [4]. In this model, the original generic non-terminal X is not substituted, rather the new non-terminals are appended as additional information to the phrases and a new feature is computed based on them. In this way the original parsing and translation spaces are left unchanged. In contrast to the above work, where the authors expand the set of linguistic non-terminals to include a large set of new symbols, we restrict ourselves to the non-terminals that are to be found in the syntax tree.

Each lexical phrase is marked with the non-terminal symbol of the best matching node as described above. When producing hierarchical rules, the gaps are labelled with the non-terminal symbols of the corresponding phrases instead of the original generic non-terminal X. It is important to point out that the syntax information is extracted from the target side only, but the substitution of the corresponding non-terminal symbol is carried out both on the source and the target sides (with the same non-terminal on both sides).

For every rule in the grammar we will store information about the possible non-terminals that can be substituted in place of the generic non-terminal X, together with a probability for each combination of non-terminal symbols. More formally, let S be the set of possible syntax non-terminals. Given a rule r with n gaps, we will define a probability distribution $p(\mathbf{s}|r)$ over S^{n+1}, where $\mathbf{s}$ denotes a possible combination of syntax non-terminal symbols to be substituted in the rule, including the left-hand-side.

We will illustrate this concept with an example. Consider the rule

$$r = X \rightarrow \left\langle uX^{\sim 0}vX^{\sim 1}w, xX^{\sim 0}yX^{\sim 1}z \right\rangle \tag{2}$$

and let $\mathbf{s} = (A, B, C)$. Then $p(\mathbf{s}|r)$ will be the probability that the rule r is interpreted as rule

$$A \rightarrow \left\langle uB^{\sim 0}vC^{\sim 1}w, xB^{\sim 0}yC^{\sim 1}z \right\rangle . \tag{3}$$

For each derivation d we will compute two probabilities. The first one will be denoted by $p_h(Y|d)$ (h for "head") and will reflect the probability that the derivation d, under consideration of the additional non-terminal symbols, has $Y \in S$ as its starting symbol. This quantity will be needed for computing the probability $p_{\text{syn}}(d)$ that the derivation conforms with the extended set of non-terminals, i.e. that we can find a derivation with the same structure but considering the new non-terminals. The negative logarithm of this last probability will then be added as a new feature to the log-linear model combination.

For the exact definition of these two quantities we will separate the case where the top rule of derivation d is a lexical phrase (in which case the derivation consists only of one rule application) and the general case where the top rule is a hierarchical one. If the top rule r of d corresponds to a lexical phrase, the probability distribution for the non-terminals for d will equal the distribution for rule r, i.e. $p_h(s|d) = p(s|r), \forall s \in S$. Given that only one rule has been applied, the derivation fully conforms with the extended set of non-terminals, thus in this case $p_{\text{syn}}(d) = 1$. For the general case of hierarchical rules, let d be a general derivation, let r be the top rule and let $d_1, \ldots, d_n$ be the sub-derivations associated with the application of rule r in derivation d. For determining if the derivation is consistent with the extended set of non-terminals we have to consider every possible substitution of non-terminals in rule r and check the probability of the n sub-derivations to have the corresponding non-terminals. More formally:

$$p_{\text{syn}}(d) = \sum_{s \in S^{n+1}} \left(p(s|r) \cdot \prod_{k=2}^{n+1} p_h(s[k]|d_{k-1}) \right) , \tag{4}$$

where the notation $[\cdot]$ is used to represent addressing the elements of a vector. The index shifting in the product in Equation 4 is due to the fact that the first element in the vector of non-terminal substitutions is the left-hand-side of the rule. Note also that although the sum is unrestricted, most of the summands will be left out due to a zero probability in the term $p(s|r)$.

The probability p_h is computed in a similar way, but the summation index is restricted only to those vectors of non-terminal substitutions where the left-hand side is the one for which we want to compute the probability. More formally:

$$p_h(Y|d) = \sum_{s \in S^{n+1} : s[1] = Y} \left(p(s|r) \cdot \prod_{k=2}^{n+1} p_h(s[k]|d_{k-1}) \right) . \tag{5}$$

In practice, the probability distributions may be renormalized in order to avoid numerical problems.

5. Poor Man's Syntax

Let us take a step back and look at the model from Section 4.3 from a distance. We can consider the rules with the same left-hand side to be a *class* of phrases, which share some common characteristics. Similarly, the non-terminals in the right-hand side represent the preferred type of rule that should be substituted in the corresponding gap. The syntax tree of the target side of the training corpus is what defines the corresponding labels. Looking at it under this viewpoint, the parsing process is little more than a sophisticated way to cluster the phrases.

In this section we will investigate a novel approach, in which we cluster the phrases with fully automatic methods, thus avoiding the need for additional syntax information in the form of a parse tree of the training data. This can be a big advantage e.g. for under-resourced languages for which no parsers might be available. It can also reduce the computational cost of the training process, although this depends

both on the parser to use, the language pair, the size of the training corpus and the clustering algorithm. In our case, the clustering took around 20 hours, while the running time for parsing can be estimated at around 2000 sentences per hour. Looking at the corpus statistics in Section 6 we can see that the training time is dramatically reduced.

We will try to mimic the phrase "clustering" of the syntax tree. The process is represented schematically in Figure 3. First we cluster the words, very much like the POS labels do, but we apply this operation on both the source and the target sides. We use the `makecls` tool [16], which is widely used as part of the alignment training procedure in statistical machine translation. Note that in this way the mapping of words will be deterministic instead of context-dependent, as is the case with part-of-speech labels.

We then go through the table of lexical phrases and substitute each word with its corresponding class. In the example of Figure 3, the source classes are denoted as "SC" and the target classes as "TC". Note that we can do this operation already on an extracted phrase table due to the deterministic word mapping. Should we want to apply a context-dependent mapping of words, we will have to perform a new phrase extraction keeping track of the associated classes. It is also worth noting that the size of the mapped phrase table will be smaller, as the result of the word mapping may join some phrases together.

On the resulting mapped table we apply a new clustering, this time on the phrase level, assigning a label to each of them. In our experiments we used the CLUTO toolkit [17] for this step. The features for the clustering are thus the mapped words as found in the previous stage.

The lexical phrases are then labelled with the corresponding class in the left-hand side of the rule. The hierarchical phrases are labelled in the same way in the left-hand side, and the gaps are labelled with the corresponding classes of the phrases that produced the gaps. This corresponds to the labelling procedure using syntactic labels described in Section 4.3. As we did there, we also considered these nonterminals as soft syntactic labels and store them as additional information associated with the rules, they do not constitute hard constraints.

6. Experimental Results

We present results on the Chinese-English NIST 2008 task. We used a selected subset of the available training material to arrive at a medium sized training corpus. The NIST 2006 was used as development set for minimum error training on BLEU in all the experiments. Table 1 shows the statistics of the data.

		Chinese	English
Train	Sentences	3 030 696	
	No. of Words	77 456 152	81 002 954
	Vocabulary	83 128	213 076
	Singletons	21 059	95 544
Dev	Sentences	1 664	
	No. of Words	42 930	172 324
	Vocabulary	6 387	17 202
	OOVs	1 871	50 353
Test	Sentences	1 357	
	No. of Words	36 114	149 057
	Vocabulary	6 418	17 877
	OOVs	1 375	43 724

Table 1: Statistics for the Chinese-English corpus

6.1. Syntactic Approaches

Our first goal was to compare the syntax-based methods with each other. Table 2 shows the results of the different approaches. The parsematch method, although it does not show any improvements on the development set, is able to improve the translation on the test set by 0.4% BLEU and 0.5% TER, although this result is not statistically significant. String-to-dependency is able to achieve a much bigger improvement in TER (1.7%), although it is only slightly better than parsematch on BLEU. The string-to-dependency experiments were carried out rescoring 100-best lists. The soft syntactic labels produce the best BLEU score, 1% over the baseline, but are not better than the string-to-dependency approach on TER.

Next we wanted to investigate if the improvements of the different methods are complementary, or if perhaps the different models address the same flaws in the baseline translation. We performed experiments with every possible pair of approaches. The BLEU score is improved in every case with respect to each of the individual approaches alone. The TER score however, does not show the same behaviour, and when combining parsematch and string-to-dependency we obtain a deterioration of 0.9% compared to string-to-dependency alone. Applying the three approaches we obtain the best TER score, with a 2% improvement over the baseline, and 1.1% improvement in BLEU. Still, the best BLEU score is obtained applying soft syntactic labels together with string-to-dependency.

The results seem to point that parsematch and dependency are a problematic pair. This can be due to the fact that parsematch, by its design, favours syntactically valid phrases, while dependency may allow for incomplete chunks, for which it remembers information about how they should be completed. We speculate that there may be conflicts when these two approaches interact and this may produce the drop in performance when measured in BLEU.

6.2. Poor man's syntax

For the poor man's syntax approach there are two additional parameters that have to be chosen, namely the number of

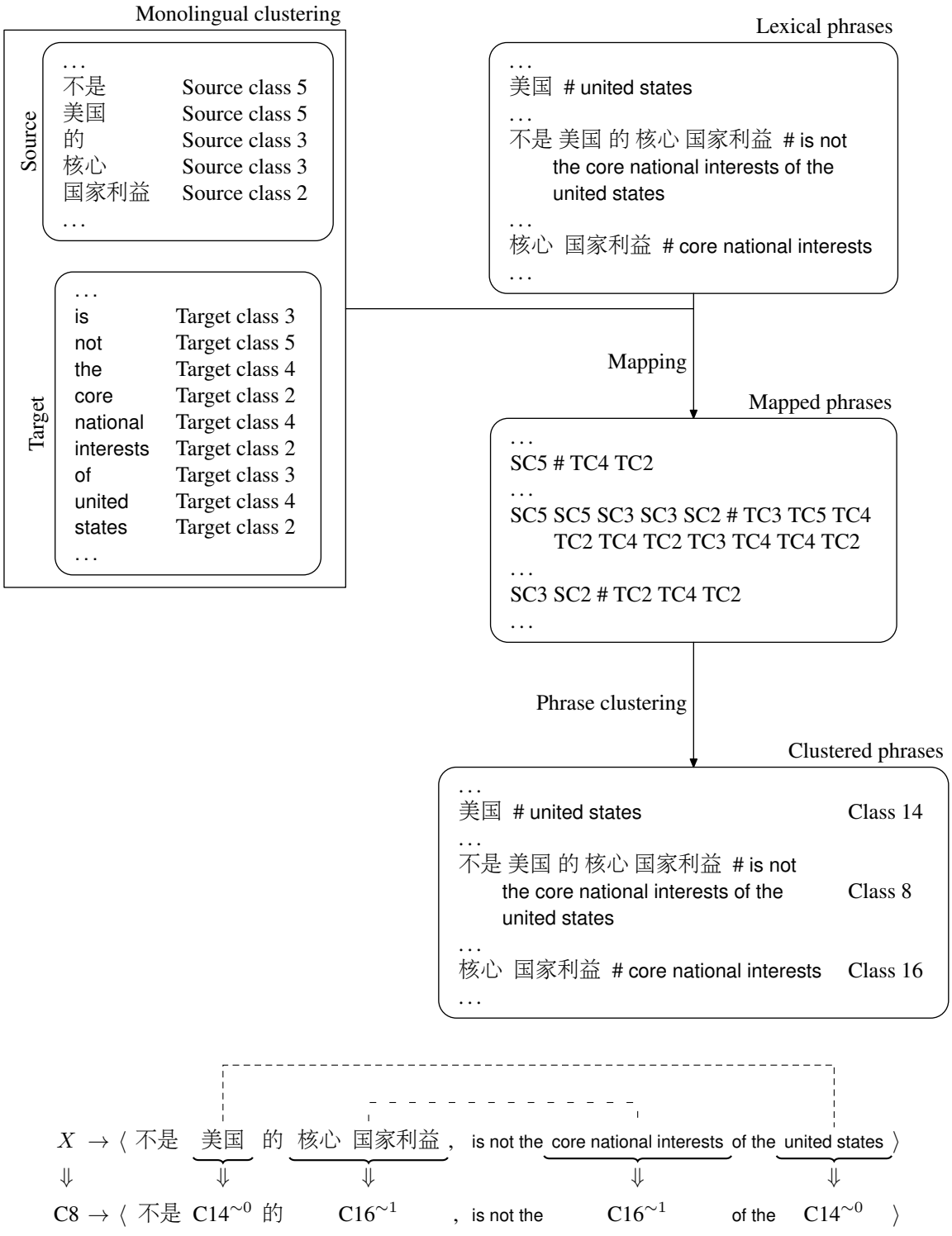

Figure 3: Illustration of the extraction procedure for the poor man's syntax method

	NIST '06 (dev)		NIST '08 (test)	
	BLEU	TER	BLEU	TER
baseline	31.4	63.2	24.0 ± 0.9	68.4 ± 0.6
parsematch	31.4	63.1	24.4	67.9
dependency	32.2	61.9	24.6	**66.7**
syntax labels	32.2	62.1	**25.0**	**67.2**
parsematch + dependency	32.0	62.5	24.6	**67.6**
syntax labels + parsematch	32.4	62.3	**25.3**	**67.3**
syntax labels + dependency	32.9	61.4	**25.4**	**66.7**
syntax labels + parsematch + dependency	32.9	61.0	**25.1**	**66.4**
poor man's syntax	32.1	62.0	24.8	**66.9**

Table 2: Results for the additional syntactic models on the NIST '06 and the NIST '08 test set. All the scores are in percentage. The 95% confidence interval is given for the baseline system. Results in bold are significantly better than the baseline.

classes the words and the phrases are to be clustered into. Each combination requires a new clustering process, a new extraction of hierarchical phrases and, for optimal results, a new run of minimum error rate training. The extracted phrase tables are also of considerable size on the hard disk, and conducting a series of experiments may easily fill up the file servers. For these reasons we carried out a non-exhaustive search for the best combination of the number of classes for the word and phrase clustering algorithms. Somehow surprisingly we arrived at a relatively low number of classes for both: 5 classes for word clustering and 20 for phrase clustering.

Table 2 shows the results obtained when applying this approach. This model achieves an improvement of 0.8% in BLEU[1] and 1.5% in TER over the baseline. This makes it comparable to the best performing syntax-based methods on both scores. In this way we are able to simulate the effect of including syntax information by applying only purely automatic methods. This is a promising result, specially for tasks where obtaining syntax information in form of parse trees is difficult or even impossible.

7. Conclusions

In this paper we have reviewed and analyzed three different possibilities for augmenting the hierarchical phrase-based translation approach including syntactic information. We have compared the performance of each of them separately and of the combination of them. We found out that the combination of different approaches further improves translation quality, which constitutes and indicator that the models address different problems in the translation process.

We have presented a new model which, while inspired by one of the syntax enhancements, does not need any additional information in the form of parse trees. This model can thus be applied to every language pair, even for under-resourced languages, for which no linguistic tools may be available. The results obtained with this approach are on-par with the syntactic models, being only slightly below the best performing systems on both the BLEU and TER scores.

The main advantage of the proposed method is that we leave it open to the algorithms how to classify the rules. When using syntax labels (the more similar method), we try to "enforce" a structure that has been defined outside of the translation task. Intuitively it is an appealing concept that the produced text should follow this structure, but it may not be the optimal for the translation task. By using automatic methods we allow for more flexibility. E.g. in this paper we allowed information about the source side to enter the rule labelling process.

There are still many directions to research for this new model. We used fairly standard clustering techniques for both words and phrases. Better clustering algorithms that are more tailored for the task at hand may further improve the results. Specifically, we ignored the context the phrases appear in, and this may prove to be relevant for a more efficient classification scheme.

[1] This result is not statistically significant at a 95% confidence level, but it is at 90%.

8. References

[1] D. Chiang, "A Hierarchical Phrase-Based Model for Statistical Machine Translation," in *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, Ann Arbor, Michigan, USA, June 2005, pp. 263–270.

[2] D. Vilar, D. Stein, and H. Ney, "Analysing Soft Syntax Features and Heuristics for Hierarchical Phrase Based Machine Translation," in *International Workshop on Spoken Language Translation*, Waikiki, Hawaii, Oct. 2008, pp. 190–197.

[3] L. Shen, J. Xu, and R. Weischedel, "A New String-

to-Dependency Machine Translation Algorithm with a Target Dependency Language Model," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Columbus, Ohio, June 2008, pp. 577–585.

[4] A. Venugopal and A. Zollmann, "Preference Grammars: Softening Syntactic Constraints to Improve Statistical Machine Translation," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, Boulder, Colorado, USA, June 2009, pp. 236–244.

[5] K. Yamada and K. Knight, "A Syntax-Based Statistical Translation Model," in *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, Toulouse, France, July 2001, pp. 523–530.

[6] S. DeNeefe, K. Knight, W. Wang, and D. Marcu, "What Can Syntax-based MT Learn from Phrase-based MT?" in *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, Prague, Czech Republic, June 2007, pp. 755–763.

[7] D. Chiang, K. Knight, and W. Wang, "11,001 new features for statistical machine translation," in *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Assoc iation for Computational Linguistics*, Boulder, Colorado, June 2009, pp. 218–226.

[8] Y. Marton and P. Resnik, "Soft Syntactic Constraints for Hierarchical Phrased-Based Translation," in *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, Columbus, Ohio, June 2008, pp. 1003–1011.

[9] D. Chiang, "Hierarchical phrase-based translation," *Computational Linguistics*, vol. 33, no. 2, pp. 201–228, June 2007.

[10] R. Zens, F. J. Och, and H. Ney, "Phrase-Based Statistical Machine Translation," in *German Conference on Artificial Intelligence*, Aachen, Germany, Sept. 2002, pp. 18–32.

[11] P. Koehn, F. J. Och, and D. Marcu, "Statistical Phrase-Based Translation," in *Proceedings of the Human Language Technology, North American Chapter of the Association for Computational Linguistics*, Edmonton, Canada, May 2003, pp. 54–60.

[12] F. J. Och and H. Ney, "Discriminative Training and Maximum Entropy Models for Statistical Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA, July 2002, pp. 295–302.

[13] F. J. Och, "Minimum Error Rate Training for Statistical Machine Translation," in *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, Sapporo, Japan, July 2003, pp. 160–167.

[14] M. Galley, M. Hopkins, K. Knight, and D. Marcu, "What's in a translation rule?" in *Proceedings of the Human Language Technology Conference / North American Chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL)*, Boston, Massachusets, USA, May 2004, pp. 273–280.

[15] A. Zollmann and A. Venugopal, "Syntax Augmented Machine Translation Via Chart Parsing," in *Proceedings of the Workshop on Statistical Machine Translation*, New York City, New York, USA, June 2006, pp. 138–141.

[16] F. J. Och, "An efficient method for determining bilingual word classes," in *Proceedings of the Ninth Conference of the European Chapter of the Association for Computational Linguistics*, June 1999, pp. 8–12.

[17] Y. Zhao and G. Karypis, "Clustering in Life Sciences," *Functional Genomics: Methods and Protocols*, vol. 224, pp. 183–218, 2003.