Improving reordering performance using higher order and structural features

Mitesh M. Khapra
IBM Research India
mikhapra@in.ibm.com

Ananthakrishnan Ramanathan
IBM Research India
anandr42@gmail.com

Karthik Visweswariah
IBM Research India
v-karthik@in.ibm.com

Abstract

Recent work has shown that word aligned data can be used to learn a model for reordering source sentences to match the target order. This model learns the cost of putting a word immediately before another word and finds the best reordering by solving an instance of the Traveling Salesman Problem (TSP). However, for efficiently solving the TSP, the model is restricted to pairwise features which examine only a pair of words and their neighborhood. In this work, we go beyond these pairwise features and learn a model to rerank the n-best reorderings produced by the TSP model using higher order and structural features which help in capturing longer range dependencies. In addition to using a more informative set of source side features, we also capture target side features indirectly by using the translation score assigned to a reordering. Our experiments, involving Urdu-English, show that the proposed approach outperforms a state-of-the-art PBSMT system which uses the TSP model for reordering by 1.3 BLEU points, and a publicly available state-of-the-art MT system, Hiero, by 3 BLEU points.

1 Introduction

Handling the differences in word orders between pairs of languages is crucial in producing good machine translation. This is especially true for language pairs such as Urdu-English which have significantly different sentence structures. For example, the typical word order in Urdu is Subject Object Verb whereas the typical word order in English is Subject Verb Object. Phrase based systems (Koehn et al., 2003) rely on a lexicalized distortion model (Al-Onaizan and Papineni, 2006; Tillman, 2004) and the target language model to produce output words in the correct order. This is known to be inadequate when the languages are very different in terms of word order (refer to Table 3 in Section 3).

Pre-ordering source sentences while training and testing has become a popular approach in overcoming the word ordering challenge. Most techniques for pre-ordering (Collins et al., 2005; Wang et al., 2007; Ramanathan et al., 2009) depend on a high quality source language parser, which means these methods work only if the source language has a parser (this rules out many languages). Recent work (Visweswariah et al., 2011) has shown that it is possible to learn a reordering model from a relatively small number of hand aligned sentences . This eliminates the need of a source or target parser.

In this work, we build upon the work of Visweswariah et al. (2011) which solves the reordering problem by treating it as an instance of the Traveling Salesman Problem (TSP). They learn a model which assigns costs to all pairs of words in a sentence, where the cost represents the penalty of putting a word immediately preceding another word. The best permutation is found via the chained Lin-Kernighan heuristic for solving a TSP. Since this model relies on solving a TSP efficiently, it cannot capture features other than pairwise features that examine the words and neighborhood for each pair of words in the source sentence. In the remainder of this paper we refer to this model as the TSP model.

Our aim is to go beyond this limitation of the TSP model and use a richer set of features instead of using pairwise features only. In particular, we are interested in features that allow us to examine triples of words/POS tags in the candidate reordering per-

Proceedings of NAACL-HLT 2013, pages 315–324,
Atlanta, Georgia, 9–14 June 2013. ©2013 Association for Computational Linguistics

mutation (this is akin to going from bigram to trigram language models), and also structural features that allow us to examine the properties of the segmentation induced by the candidate permutation. To go beyond the set of features incorporated by the TSP model, we do not solve the search problem which would be NP-hard. Instead, we restrict ourselves to an n-best list produced by the base TSP model and then search in that list. Using a richer set of features, we learn a model to rerank these n-best reorderings. The parameters of the model are learned using the averaged perceptron algorithm. In addition to using a richer set of source side features we also indirectly capture target side features by interpolating the score assigned by our model with the score assigned by the decoder of a MT system.

To justify the use of these informative features, we point to the example in Table 1. Here, the head (*driver*) of the underlined English Noun Phrase (*The driver of the car*) appears to the left of the Noun Phrase whereas the head (*chaalak* {driver}) of the corresponding Urdu Noun Phrase (*gaadi* {*car*} *ka* {*of*} *chaalak* {*driver*}) appears to the right of the Noun Phrase. To produce the correct reordering of the source Urdu sentence the model has to make an unusual choice of putting *gaadi* {car} before *bola* {said}. We say this is an unusual choice because the model examines only pairwise features and it is unlikely that it would have seen sentences having the bigram "*car said*". If the exact segmentation of the source sentence was known, then the model could have used the information that the word *gaadi* {car} appears in a segment whose head is the noun *chaalak* {driver} and hence its not unusual to put *gaadi* {car} before *bola* {said} (because the construct "*NP said*" is not unusual). However, since the segmentation of the source sentence is not known in advance, we use a heuristic (explained later) to find the segmentation induced by a reordering. We then extract features (such as $first\_word\_current\_segment$, $end\_word\_current\_segment$) to approximate these long range dependencies.

Using this richer set of features with Urdu-English as the source language pair, our approach outperforms the following state of the art systems: (i) a PBSMT system which uses TSP model for reordering (by 1.3 BLEU points), (ii) a hierarchical PBSMT system (by 3 BLEU points). The overall gain is 6.3 BLEU points when compared to a standard PBSMT system which uses a lexicalized distortion model (Al-Onaizan and Papineni, 2006).

Input Urdu:	fir gaadi ka chaalak kuch bola
Gloss:	then car of driver said something
English:	Then the driver of the car said something.
Ref. reordering:	fir chaalak ka gaadi bola kuch

Table 1: Example motivating the use of structural features

The rest of this paper is organized as follows. In Section 2 we discuss our approach of re-ranking the n-best reorderings produced by the TSP model. This includes a discussion of the model used, the features used and the algorithm used for learning the parameters of the model. It also includes a discussion on the modification to the Chained Lin-Kernighan heuristic to produce n-best reorderings. Next, in Section 3 we describe our experimental setup and report the results of our experiments. In Section 4 we present some discussions based on our study. In section 5 we briefly describe some prior related work. Finally, in Section 6, we present some concluding remarks and highlight possible directions for future work.

2 Re-ranking using higher order and structural features

As mentioned earlier, the TSP model (Visweswariah et al., 2011) looks only at local features for a word pair (w_i, w_j). We believe that for better reordering it is essential to look at higher order and structural features (*i.e.*, features which look at the overall structure of a sentence). The primary reason why Visweswariah et al. (2011) consider only pairwise bigram features is that with higher order features the reordering problem can no longer be cast as a TSP and hence cannot be solved using existing efficient heuristic solvers. However, we do not have to deal with an NP-Hard search problem because instead of considering all possible reorderings we restrict our search space to only the n-best reorderings produced by the base TSP model. Formally, given a set of reorderings, $\Pi = [\pi_1, \pi_2, \pi_3, ..., \pi_n]$, for a source sentence s, we are interesting in assigning a score, $score(\pi)$, to each of these reorderings and pick the reordering which has the highest score. In this paper, we parametrize this score as:

$$score(\pi) = \theta^T \phi(\pi) \quad (1)$$

where, θ is the weight vector and $\phi(\pi)$ is a vector of features extracted from the reordering π. The aim then is to find,

$$\pi^* = \arg\max_{\pi \in \Pi} \; score(\pi) \quad (2)$$

In the following sub-sections, we first briefly describe our overall approach towards finding π^*. Next, we describe our modification to the Lin-Kernighan heuristic for producing n-best outputs for TSP instead of the 1-best output used by (Visweswariah et al., 2011). We then discuss the features used for re-ranking these n-best outputs, followed by a discussion on the learning algorithm used for estimating the parameters of the model. Finally, we describe how we interpolate the score assigned by our model with the score assigned by the decoder of a SMT engine to indirectly capture target side features.

2.1 Overall approach

The training stage of our approach involves two phases : (i) Training a TSP model which will be used to generate n-best reorderings and (ii) Training a re-ranking model using these n-best reorderings. For training both the models we need a collection of sentences where the desired reordering $\pi^*(x)$ for each input sentence x is known. These reference orderings are derived from word aligned source-target sentence pairs (see first 4 rows of Figure 1). We first divide this word aligned data into N parts and use A^{-i} to denote the alignments leaving out the i-th part. We then train a TSP model M^{-i} using reference reorderings derived from A^{-i} as described in (Visweswariah et al., 2011). Next, we produce n-best reorderings for the source sentences using the algorithm $getNBestReorderings(sentence)$ described later. Dividing the data into N parts is necessary to ensure that the re-ranking model is trained using a realistic n-best list rather than a very optimistic n-best list (which would be the case if part i is reordered using a model which has already seen part i during training).

Each of the n-best reorderings is then represented as a feature vector comprising of higher order and structural features. The weights of these features are then estimated using the averaged perceptron method. At test time, $getNBestReorderings(sentence)$ is used to generate the n-best reorderings for the test sentence using the trained TSP model. These reorderings are then represented using higher order and structural features and re-ranked using the weights learned earlier. We now describe the different stages of our algorithm.

2.2 Generating n-best reorderings for the TSP model

The first stage of our approach is to train a TSP model and generate n-best reorderings using it. The decoder used by Visweswariah et al. (2011) relies on the Chained Lin-Kernighan heuristic (Lin and Kernighan, 1973) to produce the 1-best permutation for the TSP problem. Since our algorithm aims at re-ranking an n-best list of permutations (reorderings), we made a modification to the Chained Lin-Kernighan heuristic to produce this n-best list as shown in Algorithm 1 .

Algorithm 1 $getNBestReorderings(sentence)$

$NbestSet = \phi$
$\pi^* =$ Identity permutation
$\pi^* = linkernighan(\pi^*)$
$insert(NbestSet, \pi^*)$
for $i = 1 \rightarrow nIter$ **do**
  $\pi^{'} = perturb(\pi^*)$
  $\pi^{'} = linkernighan(\pi^{'})$
  if $C(\pi^{'}) < max_{\pi \in NbestSet} C(\pi)$ **then**
    $InsertOrReplace(NbestSet, \pi^{'})$
  end if
  if $C(\pi^{'}) < C(\pi^*)$ **then**
    $\pi^* = \pi^{'}$
  end if
end for

In Algorithm 1 $perturb()$ is a four-edge perturbation described in (Applegate et al., 2003), and $linkernighan()$ is the Lin-Kernighan heuristic that applies a sequence of flips that potentially returns a lower cost permutation as described in (Lin and Kernighan, 1973). The cost $C(\pi)$ is calculated using a trained TSP model.

2.3 Features

We represent each of the n-best reorderings obtained above as a vector of features which can be divided into two sets : (i) higher order features and (ii) struc-

Segmentation Based Features (extracted for every segment in the induced segmentation)	Features fired for the segment [mere(PRP) ghar(NN)] in Figure1
$end\_lex\_current\_segment$	$ghar$
$end\_lex\_prev\_segment$	$Shyam$
$end\_pos\_current\_segment$	NN
$end\_pos\_prev\_segment$	NN
$length\_of\_current\_segment$	2
$first\_lex\_current\_segment$	$mere$
$first\_lex\_next\_segment$	$aaye$
$first\_pos\_current\_segment$	PRP
$first\_pos\_next\_segment$	VRB
Higher order features	**Features fired for the triplet Shyam(NN) the(Vaux) aaye(VRB) in Figure1**
$lex\_triplet\_jumps$	$lex\_triplet$ = *"Shyam the aaye"* && *jumps* $= [4, -1]$
$pos\_triplet\_jumps$	$pos\_triplet$ = *"NN Vaux VRB"* && *jumps* $= [4, -1]$

Table 2: Features used in our model.

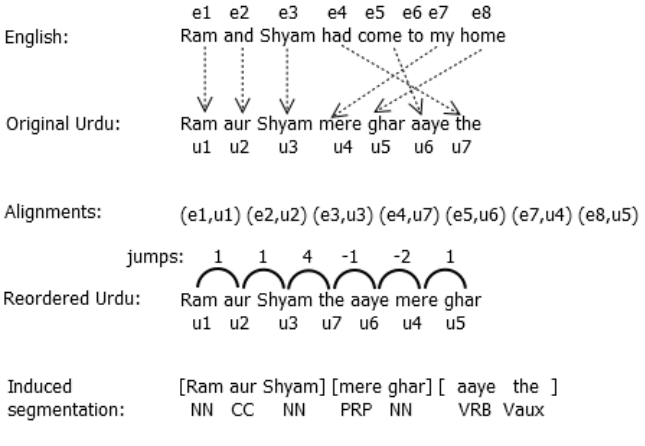

Figure 1: Segmentation induced on the Urdu sentence when it is reordered according to its English translation. Note that the words *Shyam* and *mere* are adjacent to each other in the original Urdu sentence but not in the reordered Urdu sentence. Hence, the word *mere* marks the beginning of a new segment.

tural features. The higher order features are essentially trigram lexical and pos features whereas the structural features are derived from the sentence structure induced by a reordering (explained later).

2.3.1 Higher Order Features

Since deriving a good reordering would essentially require analyzing the syntactic structure of the source sentence, the tasks of reordering and parsing are often considered to be related. The main motivation for using higher order features thus comes from a related work on parsing (Koo and Collins, 2010) where the performance of a state of the art parser was improved by considering higher order dependencies. In our model we use trigram features (see Table 2) of the following form:

$\phi(ru_i, ru_{i+1}, ru_{i+2}, J(ru_i, ru_{i+1}), J(ru_{i+1}, ru_{i+2}))$
where ru_i =*word at position* i *in the reordered source sentence* and $J(x, y)$ = *difference between the positions of* x *and* y *in the original source sentence*.

Figure 1 shows an example of jumps between different word pairs in an Urdu sentence. Since such higher order features will typically be sparse, we also use some back-off features. For example, instead of using the absolute values of jumps we divide the jumps into 3 buckets, *viz.*, high, low and medium and use these buckets in conjunction with the triplets as back-off features.

2.3.2 Structural Features

The second set of features is based on the hypothesis that any reordering of the source sentence induces a segmentation on the sentence. This segmentation is based on the following heuristic: if w_i and w_{i+1} appear next to each other in the original sentence but do not appear next to each other in the reordered sentence then w_i marks the end of a segment and w_{i+1} marks the beginning of the next segment. To understand this better please refer to Figure 1 which shows the correct reordering of an Urdu sentence based on its English translation and the corresponding segmentation induced on the Urdu sentence. If the correct segmentation of a sentence is known in advance then one could use a hierarchical model where the goal would be to reorder segments instead of reordering words individually (basically, instead of words, treat segments as units of reordering. In principle, this is similar to what is done by parser based reordering methods). Since the TSP model does not explicitly use segmentation based features it often produces wrong reorderings (refer to the motivating example in Section 1).

Reordering such sentences correctly requires some knowledge about the hierarchical structure of the sentence. To capture such hierarchical information, we use features which look at the elements

(words, pos tags) of a segment and its neighboring segments. These features along with examples are listed in Table 2. These features should help us in selecting a reordering which induces a segmentation which is closest to the correct segmentation induced by the reference reordering. Note that every feature listed in Table 2 is a binary feature which takes on the value 1 if it fires for the given reordering and value 0 if it does not fire for the given reordering. In addition to the features listed in Table 2 we also use the score assigned by the TSP model as a feature.

2.4 Estimating model parameters

We use perceptron as the learning algorithm for estimating the parameters of our model described in Equation 1. To begin with, all parameters are initialized to 0 and the learning algorithm is run for N iterations. During each iteration the parameters are updated after every training instance is seen. For example, during the i-th iteration, after seeing the j-th training sentence, we update the k-th parameter θ_k using the following update rule:

$$\theta_k^{(i,j)} = \theta_k^{(i,j-1)} + \phi_k(\pi_j^{gold}) - \phi_k(\pi_j^*) \quad (3)$$

where, $\theta_k^{(i,j)}$ = *value of the k-th parameter after seeing sentence j in iteration i*

ϕ_k = *k-th feature*

π_j^{gold} = *gold reordering for the j-th sentence*

$$\pi_j^* = \arg\max_{\pi \in \Pi_j} \theta^{(i,j-1)^T} \phi(\pi)$$

where Π_j is the set of n-best reorderings for the j-th sentence. π_j^* is thus the highest-scoring reordering for the j-th sentence under the current parameter vector. Since the averaged perceptron method is known to perform better than the perceptron method, we used the averaged values of the parameters at the end of N iterations, calculated as:

$$\theta_k^{avg} = \frac{1}{N \cdot t} \sum_{i=1}^{N} \sum_{j=1}^{t} \theta_k^{(i,j)} \quad (4)$$

where, N = *Number of iterations*

t = *Number of training instances*

We observed that in most cases the reference reordering in not a part of the n-best list produced by the TSP model. In such cases instead of using $\phi_k(\pi_j^{gold})$ for updating the weights in Equation 3 we use $\phi_k(\pi_j^{closest\_to\_gold})$ as this is known to be a better strategy for learning a re-ranking model (Arun and Koehn, 2007). $\pi_j^{closest\_to\_gold}$ is given by:

$$\arg\max_{\pi_j^i \in \Pi_j} \frac{\textit{\# of common bigram pairs in } \pi_j^i \textit{ and } \pi_j^{gold}}{len(\pi_j^{gold})}$$

where, Π_j = *set of n-best reorderings for j^{th} sentence*

$\pi_j^{closest\_to\_gold}$ is thus the reordering which has the maximum overlap with π_j^{gold} in terms of the number of word pairs (w_m, w_n) where w_n is put next to w_m.

2.5 Interpolating with MT score

The approach described above aims at producing a better reordering by extracting richer features from the source sentence. Since the final aim is to improve the performance of an MT system, it would potentially be beneficial to interpolate the scores assigned by Equation 1 to a given reordering with the score assigned by the decoder of an MT system to the translation of the source sentence under this reordering. Intuitively, the MT score would allow us to capture features from the target sentence which are obviously not available to our model. With this motivation, we use the following interpolated score ($score_I$) to select the best translation.

$$score_I(t_i) = \lambda \cdot score_\theta(\pi_i) + (1 - \lambda) \cdot score_{MT}(t_i)$$

where, t_i = *translation produced under the i-th reordering of the source sentence*

$score_\theta(\pi_i)$ = *score assigned by our model to the i-th reordering*

$score_{MT}(t_i)$ = *score assigned by the MT system to* t_i

The weight λ is used to ensure that $score_\theta(\pi_i)$ and $score_{MT}(\pi_i)$ are in the same range (it just serves as a normalization constant). We acknowledge that the above process is expensive because it requires the MT system to decode n reorderings for every source sentence. However, the aim of this work is to show that interpolating with the MT score which implicitly captures features from the target sentence helps in improving the performance. Ideally, this interpolation should (and can) be done at decode time without having to decode n reorderings for every source

sentence (for example by expressing the n reorderings as a lattice), but, we leave this as future work.

3 Empirical evaluation

We evaluated our reordering approach on Urdu-English. We use two types of evaluation, one intrinsic and one extrinsic. For intrinsic evaluation, we compare the reordered source sentence in Urdu with a reference reordering obtained from the hand alignments using BLEU (referred to as monolingual BLEU or mBLEU by (Visweswariah et al., 2011)). Additionally, we evaluate the effect of reordering on MT performance using BLEU (extrinsic evaluation).

As mentioned earlier, our training process involves two phases : (i) Generating n-best reorderings for the training data and (ii) using these n-best reorderings to train a perceptron model. We use the same data for training the reordering model as well as our perceptron model. This data contains 180K words of manual alignments (part of the NIST MT-08 training data) and 3.9M words of automatically generated machine alignments (1.7M words from the NIST MT-08 training data[1] and 2.2M words extracted from sources on the web[2]). The machine alignments were generated using a supervised maximum entropy model (Ittycheriah and Roukos, 2005) and then corrected using an improved correction model (McCarley et al., 2011). We first divide the training data into 10 folds. The n-best reorderings for each fold are then generated using a model trained on the remaining 9 folds. This division into 10 folds is done for reasons explained earlier in Section 2.1. These n-best reorderings are then used to train the perceptron model as described in Section 2.4. Note that Visweswariah et al. (2011) used only manually aligned data for training the TSP model. However, we use machine aligned data in addition to manually aligned data for training the TSP model as it leads to better performance. We used this improvised TSP model as the state of the art baseline (rows 2 and 3 in Tables 3 and 4 respectively) for comparing with our approach.

We observed that the perceptron algorithm converges after 5 iterations beyond which there is very little ($<1\%$) improvement in the bigram precision on the training data itself (bigram precision is the fraction of word pairs which are correctly put next to each other). Hence, for all the numbers reported in this paper, we used 5 iterations of perceptron training. Similarly, while generating the n-best reorderings, we experimented with following values of n : 10, 25, 50, 100 and 200. We observed that, by restricting the search space to the top-50 reorderings we get the best reordering performance (mBLEU) on a development set. Hence, we used n=50 for our MT experiments.

For intrinsic evaluation we use a development set of 8017 Urdu tokens reordered manually. Table 3 compares the performance of the top-1 reordering output by our algorithm with the top-1 reordering generated by the improved TSP model in terms of mBLEU. We see a gain of 1.8 mBLEU points with our approach.

Next, we see the impact of the better reorderings produced by our system on the performance of a state-of-the-art MT system. For this, we used a standard phrase based system (Al-Onaizan and Papineni, 2006) with a lexicalized distortion model with a window size of +/-4 words (Tillmann and Ney, 2003). As mentioned earlier, our training data consisted of 3.9M words including the NIST MT-08 training data. We use HMM alignments along with higher quality alignments from a supervised aligner (McCarley et al., 2011). The Gigaword English corpus was used for building the English language model. We report results on the NIST MT-08 evaluation set, averaging BLEU scores from the News and Web conditions to provide a single BLEU score. Table 4 compares the MT performance obtained by reordering the training and test data using the following approaches:

1. No pre-ordering: A baseline system which does not use any source side reordering as a pre-processing step

2. HIERO : A state of the art hierarchical phrase based translation system (Chiang, 2007)

3. TSP: A system which uses the 1-best reordering produced by the TSP model

4. Higher order & structural features: A system

[1] http://www.ldc.upenn.edu

[2] http://centralasiaonline.com

Approach	mBLEU
Unreordered	31.2
TSP	56.6
Higher order & structural features	**58.4**

Table 3: mBLEU scores for Urdu to English reordering using different models.

Approach	BLEU
No pre-ordering	21.9
HIERO	25.2
TSP	26.9
Higher order & structural features	27.5
Interpolating with MT score	**28.2**

Table 4: MT performance for Urdu to English without reordering and with reordering using different approaches.

which reranks n-best reorderings produced by TSP using higher order and structural features

5. Interpolating with MT score : A system which interpolates the score assigned to a reordering by our model with the score assigned by a MT system

We used Joshua 4.0 (Ganitkevitch et al., 2012) which provides an open source implementation of HIERO. For training, tuning and testing HIERO we used the same experimental setup as described above. As seen in Table 4, we get an overall gain of 6.2 BLEU points with our approach as compared to a baseline system which does not use any reordering. More importantly, we outperform (i) a PBSMT system which uses the TSP model by 1.3 BLEU points and (ii) a state of the art hierarchical phrase based translation system by 3 points.

4 Discussions

We now discuss some error corrections and ablation tests.

4.1 Example of error correction

We first give an example where the proposed approach performed better than the TSP model. In the example below, *I* = input sentence, *E*= gold English translation, *T* = incorrect reordering produced by TSP and *O* = correct reordering produced by our approach. Note that the words *roman catholic aur protestant* in the input sentence get translated as a continuous phrase in English (*Roman Catholic and Protestant*) and hence should be treated as a single unit by the reordering model. The TSP model fails to keep this segment intact whereas our model (which uses segmentation based features) does so and matches the reference reordering.

Sentence length	mBLEU		
	Unreordered	TSP	Our approach
1-14 words (small)	29.7	58.7	57.8
15-22 words (med.)	28.2	56.8	59.2
23+ words (long)	33.4	55.8	58.2
All	31.2	56.6	58.4

Table 5: mBLEU improvements on sentences of different lengths

I: ab roman catholic aur protestant ke darmiyaan ikhtilafat khatam ho chuke hai

E: The differences between Roman Catholics and Protestants have now ended

T: ab roman ikhtilafat ke darmiyaan catholic aur protestant hai khatam ho chuke

O: ab ikhtilafat ke darmiyaan roman catholic aur protestant hai khatam ho chuke

4.2 Performance based on sentence length

We split the test data into roughly three equal parts based on length, and calculated the mBLEU improvements on each of these parts as reported in Table 5. These results show that the model works much better for medium-to-long sentences. In fact, we see a drop in performance for small sentences. A possible reason for this could be that the structural features that we use are derived through a heuristic that is error-prone, and in shorter sentences, where there would be fewer reordering problems, these errors hurt more than they help. While this needs to be analyzed further, we could meanwhile combine the two models fruitfully by using the base TSP model for small sentences and the new model for longer sentences.

Disabled feature	mBLEU
$end\_lex\_current\_segment$	57.6
$end\_lex\_prev\_segment$	57.6
$end\_pos\_current\_segment$	57.8
$end\_pos\_prev\_segment$	57.4
$length$	57.6
$lex\_triplet\_jumps$	58.0
$pos\_triplet\_jumps$	56.1
$first\_lex\_current\_segment$	58.2
$first\_lex\_next\_segment$	58.2
$first\_pos\_current\_segment$	57.6
$first\_pos\_next\_segment$	57.6
$NONE$	58.4

Table 6: Ablation test indicating the contribution of each feature to the reordering performance.

4.3 Ablation test

To study the contribution of each feature to the reordering performance, we did an ablation test wherein we disabled one feature at a time and measured the change in the mBLEU scores. Table 6 summarizes the results of our ablation test. The maximum drop in performance is obtained when the $pos\_triplet\_jumps$ feature is disabled. This observation supports our claim that higher order features (more than bigrams) are essential for better reordering. The $lex\_triplet\_jumps$ feature has the least impact on the performance mainly because it is a lexicalized feature and hence very sparse. Also note that there is a high correlation between the performances obtained by dropping one feature from each of the following pairs :
i) *first_lex_current_segment*, *first_lex_next_segment*
ii) *first_pos_current_segment*, *first_pos_next_segment*
iii) *end_lex_current_segment*, *end_lex_next_segment*.
This is because these pairs of features are highly dependent features. Note that similar to the $pos\_triplet\_jumps$ feature we also tried a $pos\_quadruplet\_jumps$ feature but it did not help (mainly due to overfitting and sparsity).

5 Related Work

There are several studies which have shown that reordering the source side sentence to match the target side order leads to improvements in Machine Translation. These approaches can be broadly classified into three types. First, approaches which reorder source sentences by applying rules to the source side parse; the rules are either hand-written (Collins et al., 2005; Wang et al., 2007; Ramanathan et al., 2009) or learned from data (Xia and McCord, 2004; Genzel, 2010; Visweswariah et al., 2010). These approaches require a source side parser which is not available for many languages. The second type of approaches treat machine translation decoding as a parsing problem by using source and/or target side syntax in a Context Free Grammar framework. These include Hierarchical models (Chiang, 2007) and syntax based models (Yamada and Knight, 2002; Galley et al., 2006; Liu et al., 2006; Zollmann and Venugopal, 2006). The third type of approaches, avoid the use of a parser (as required by syntax based models) and instead train a reordering model using reference reorderings derived from aligned data. These approaches (Tromble and Eisner, 2009; Visweswariah et al., 2011; DeNero and Uszkoreit, 2011; Neubig et al., 2012) have a low decode time complexity as reordering is done as a preprocessing step and not integrated with the decoder.

Our work falls under the third category, as it improves upon the work of (Visweswariah et al., 2011) which is closely related to the work of (Tromble and Eisner, 2009) but performs better. The focus of our work is to use higher order and structural features (based on segmentation of the source sentence) which are not captured by their model. Some other works have used collocation based segmentation (Henríquez Q. et al., 2010) and Multiword Expressions as segments (Bouamor et al., 2012) to improve the performance of SMT but without much success. The idea of improving performance by reranking a n-best list of outputs has been used recently for the related task of parsing (Katz-Brown et al., 2011) using targeted self-training for improving the performance of reordering. However, in contrast, in our work we directly aim at improving the performance of a reordering model.

6 Conclusion

In this work, we proposed a model for re-ranking the n-best reorderings produced by a state of the art reordering model (TSP model) which is limited to pair wise features. Our model uses a more informative set of features consisting of higher order features, structural features and target side features

(captured indirectly using translation scores). The problem of intractability is solved by restricting the search space to the n-best reorderings produced by the TSP model. A detailed ablation test shows that of all the features used, the pos triplet features are most informative for reordering. A gain of 1.3 and 3 BLEU points over a state of the art phrase based and hierarchical machine translation system respectively provides good extrinsic validation of our claim that such long range features are useful.

As future work, we would like to evaluate our algorithm on other language pairs. We also plan to integrate the score assigned by our model into the decoder to avoid having to do n decodings for every source sentence. Also, it would be interesting to model the segmentation explicitly, where the aim would be to first segment the sentence and then use a two level hierarchical reordering model which first reorders these segments and then reorders the words within the segment.

References

Yaser Al-Onaizan and Kishore Papineni. 2006. Distortion models for statistical machine translation. In *Proceedings of ACL*, ACL-44, pages 529–536, Morristown, NJ, USA. Association for Computational Linguistics.

David Applegate, William Cook, and Andre Rohe. 2003. Chained lin-kernighan for large traveling salesman problems. In *INFORMS Journal On Computing*.

Abhishek Arun and Philipp Koehn. 2007. Online learning methods for discriminative training of phrase based statistical machine translation. In *In Proceedings of MT Summit*.

Dhouha Bouamor, Nasredine Semmar, and Pierre Zweigenbaum. 2012. Identifying bilingual multiword expressions for statistical machine translation. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uur Doan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may. European Language Resources Association (ELRA).

David Chiang. 2007. Hierarchical phrase-based translation. *Comput. Linguist.*, 33(2):201–228, June.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of ACL*, pages 531–540, Morristown, NJ, USA. Association for Computational Linguistics.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 193–203, Stroudsburg, PA, USA. Association for Computational Linguistics.

Michel Galley, Jonathan Graehl, Kevin Knight, Daniel Marcu, Steve DeNeefe, Wei Wang, and Ignacio Thayer. 2006. Scalable inference and training of context-rich syntactic translation models. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 961–968, Stroudsburg, PA, USA. Association for Computational Linguistics.

Juri Ganitkevitch, Yuan Cao, Jonathan Weese, Matt Post, and Chris Callison-Burch. 2012. Joshua 4.0: Packing, pro, and paraphrases. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 283–291, Montréal, Canada, June. Association for Computational Linguistics.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, COLING '10, pages 376–384, Stroudsburg, PA, USA. Association for Computational Linguistics.

A. Carlos Henríquez Q., R. Marta Costa-jussà, Vidas Daudaravicius, E. Rafael Banchs, and B. José Mariño. 2010. Using collocation segmentation to augment the phrase table. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, WMT '10, pages 98–102, Stroudsburg, PA, USA. Association for Computational Linguistics.

Abraham Ittycheriah and Salim Roukos. 2005. A maximum entropy word aligner for Arabic-English machine translation. In *Proceedings of HLT/EMNLP*, HLT '05, pages 89–96, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jason Katz-Brown, Slav Petrov, Ryan McDonald, Franz Och, David Talbot, Hiroshi Ichikawa, Masakazu Seno, and Hideto Kazawa. 2011. Training a parser for machine translation reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 183–192, Stroudsburg, PA, USA. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of HLT-NAACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proceedings of the 48th*

Annual Meeting of the Association for Computational Linguistics, ACL '10, pages 1–11, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. Lin and B. W. Kernighan. 1973. An effective heuristic algorithm for the travelling-salesman problem. *Operations Research*, pages 498–516.

Yang Liu, Qun Liu, and Shouxun Lin. 2006. Tree-to-string alignment template for statistical machine translation. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 609–616, Stroudsburg, PA, USA. Association for Computational Linguistics.

J. Scott McCarley, Abraham Ittycheriah, Salim Roukos, Bing Xiang, and Jian-ming Xu. 2011. A correction model for word alignments. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 889–898, Stroudsburg, PA, USA. Association for Computational Linguistics.

Graham Neubig, Taro Watanabe, and Shinsuke Mori. 2012. Inducing a discriminative parser to optimize machine translation reordering. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 843–853, Jeju Island, Korea, July. Association for Computational Linguistics.

Ananthakrishnan Ramanathan, Hansraj Choudhary, Avishek Ghosh, and Pushpak Bhattacharyya. 2009. Case markers and morphology: addressing the crux of the fluency problem in English-Hindi smt. In *Proceedings of ACL-IJCNLP*.

Christoph Tillman. 2004. A unigram orientation model for statistical machine translation. In *Proceedings of HLT-NAACL*.

Christoph Tillmann and Hermann Ney. 2003. Word reordering and a dynamic programming beam search algorithm for statistical machine translation. *Computational Linguistics*, 29(1):97–133.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of EMNLP*.

Karthik Visweswariah, Jiri Navratil, Jeffrey Sorensen, Vijil Chenthamarakshan, and Nandakishore Kambhatla. 2010. Syntax based reordering with automatically derived rules for improved statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*.

Karthik Visweswariah, Rajakrishnan Rajkumar, Ankur Gandhe, Ananthakrishnan Ramanathan, and Jiri Navratil. 2011. A word reordering model for improved machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 486–496, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of EMNLP-CoNLL*.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *COLING*.

Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical mt. In *Proceedings of ACL*.

Andreas Zollmann and Ashish Venugopal. 2006. Syntax augmented machine translation via chart parsing. In *Proceedings on the Workshop on Statistical Machine Translation*.