Iterative Reordering and Word Alignment for Statistical MT

Sara Stymne
Department of Computer and Information Science
Linköpings universitet, Linköping, Sweden
sara.stymne@liu.se

Abstract

Word alignment is necessary for statistical machine translation (SMT), and reordering as a preprocessing step has been shown to improve SMT for many language pairs. In this initial study we investigate if both word alignment and reordering can be improved by iterating these two steps, since they both depend on each other. Overall no consistent improvements were seen on the translation task, but the reordering rules contain different information in the different iterations, leading us to believe that the iterative strategy can be useful.

1 Introduction

Reordering is a problem for translation between languages with different word order, such as English and German, where especially the placement of verbs vary widely. A common strategy for approaches that tackle reordering differences in connection with SMT is to perform reordering of the source language corpus prior to training the system, in order to make the word order more similar to that of the target language. Some of the reordering strategies proposed uses word alignments between texts as their main knowledge source for learning reorderings. Word alignments are also created automatically with methods that perform better when the word order in the two languages is similar. This leads us to the hypothesis that we should be able to improve both reordering rules and word alignments by performing the two tasks iteratively.

2 Previous work

Pre-translation reordering is usually performed by applying rules, that can either be handwritten rules targeting known syntactic differences (Collins et al., 2005), or be learnt automatically (Xia and McCord, 2004). In these studies the reordering decision was taken deterministically on the source side. This decision can also be delayed to decoding time by presenting several reordering options to the decoder as a lattice (Rottmann and Vogel, 2007). There have also been attempts to integrate reordering rules into a PBSMT decoder (Elming, 2008). A different way to use reordering, was investigated in Holmqvist et al. (2009), who used the reordering only to improve the word alignment, and moved the words back into original order after the alignment phase. For most of the automatically learnt rules, some rule-extraction method is used, that only takes the word alignments into account.

This work is inspired by the approach of Holmqvist et al. (2009), but further develops it both by iterating word alignment and reordering, and by creating rules that can be used on monolingual test data. The machine learning used in this study is similar to that of Elming (2008), who also uses Ripper. A different feature set is used, however. The rules are employed in a single preprocessing step, choosing the one best reordering for each sentence, similar to Xia and McCord (2004).

3 Iterative Alignment and Reordering

The steps performed in the iterative word alignment and reordering are:

1. Word align the training data
2. Learn reordering rules based on the word alignments
3. Reorder the training data with the learnt rules
4. Word align the reordered data
5. Change the order back into original order, and adjust the newly learnt word alignments
6. Learn new reordering rules based on the new word alignments
7. Repeat step 3-6

Bolette Sandford Pedersen, Gunta Nešpore and Inguna Skadiņa (Eds.)
NODALIDA 2011 Conference Proceedings, pp. 315–318

Type	LC	LS	RS	RC
Word	able	to refuse	new tasks	if
POS	A	INFMARK> V	A N	CS
Dep	comp_V	pm_V mod_A	attr_N obj_V	pm_V
Func	PCOMPL-S	INFMARK> -FMAINV	A> OBJ	CS
Syntax	NH	VG	NP	CS

Table 1: Example of a positive training example

Any automatic method can be used for word alignment. Learning reordering rules should be based on the word alignments as a knowledge source, for the iterations to be useful. In this particular implementation of the main strategy we use the standard IBM models up to model 4, as implemented in GIZA++ (Och and Ney, 2003) for word alignment, and a rule induction learner for the reordering rule learning.

4 Reordering

We used rule-induction learning, as implemented in Ripper (Cohen, 1995). A rule-induction learner produces rules for the positive class(es), where each rule only contains a subset of all features. Some example rules can be seen in Table 3. The rules are human readable, allowing us to analyse them in a useful manner, and to apply them to unseen source text in a simple way.

The reordering rules were learnt for the English source side of the corpus, which was parsed using a commercial functional dependency parser[1], from which we extracted information on the following levels: words, POS-tags, dependency information, functional tag and surface syntax. The different levels of annotation for each word, allows the learner to learn rules of different generalisation levels, possibly mixing higher-order categories with surface form in the same rule.

The reorderings were considered between two consecutive sequences, left sequence, LS, and right sequence, RS, taking the left and right one word context (LC and RC) into account. LS and RS are limited to maximum 10 tokens, and we only extract the maximum sequences for each possible reordering. Only swaps, i.e. cases where LS and RS are consecutive are considered, following Elming (2008). For each of these four sequences and contexts we stored information for each of the five syntactic levels, resulting in a total of 20 features, as exemplified in Table 1. For rules where one of LS or RS are at least 3 words long, we also store a wild card version of the example, where the middle words are replaced by an asterisk. A training example was considered positive (Swap) if the rightmost alignment point of RS is directly preceding the leftmost alignment point of LS, and negative (NoSwap) otherwise.

[1]Connexor machinese syntax, `http://www.connexor.eu/`

Orig	I(PRON,subj_V) would(+FAUXV) therefore once more ask you to ensure that we(SUBJ) get(V) a Dutch channel as well .(PUNC)
Reo	I **therefore** *would* once more ask you to ensure that we **a Dutch channel as well** *get* .
German	**Deshalb** *möchte* ich sie nochmals ersuchen , dafür sorge zu tragen , daß **auch ein niederländischer Sender** *eingespeist wird* .

Table 2: Sample rule application (Only annotations relevant to rule application are shown)

To apply the rules created by Ripper, we did not actually use Ripper, since the examples are created based on word alignments, that are not available at test time. Instead we applied a left-to-right matching directly on parsed text. All rules were applied to each sentence by first finding a matching left context, then in turn a consecutive matching left sequence, right sequence, and right context. Many of the rules, however, did not contain all these sequences, and in those cases we allowed a word sequence of up to seven words to match for LS and RS. To to be able to apply the rules safely, rules that did not either contain either both LS and RS, or one of those and both LC and RC were discarded.

In the first step a lattice was created containing all matching reorderings in a sentence, where each edge was weighted with the Ripper accuracy of the rule for the first application point of a rule, and by a small constant for all other edges. The 1-best path through the lattice was found by normalizing the scores of the outgoing edges of each node, and multiplying the normalized scores for each path, choosing the path with the highest score. Table 2 shows a sentence after application of rules (a,c) from Table 3, resulting in a word order closer to German than the original English sentence.

ID	Iter	Acc	LC	LS	RS	RC
a	1	0.82	Func:SUBJ	POS:V	–	POS:PUNC
b	1	0.73	Syntax:NH	word:could	–	Syntax:EH
c	1	0.88	POS:PRON Dep:subj_V	Func +FAUXV	word:therefore	–
d	2	0.68	Syntax:>N POS:DET	Syntax:NP POS:N	Syntax:VP	–
e	2	0.86	Func:A> POS:DET	Syntax:NP POS:N	Syntax:'VP * NH'	POS:PUNC

Table 3: Sample rules from both iterations, with Ripper accuracy

Iteration	Training		Test	
	Swap	NoSwap	Swap	NoSwap
1	689200	5974222	172084	1495241
2	648527	5827045	162533	1457786

Table 4: Reordering training/test data per iteration

5 Experiments

The experiments were performed on English-to-German translation using a standard phrase-based SMT system, trained using the Moses toolkit (Koehn et al., 2007), with a 5-gram language model. The SMT system used a distance-based reordering penalty (distortion penalty), which adds a factor δ^n for movements over n words, where phrase movement is also limited to a distance of six words. In addition we applied a monotone model, which prohibits any phrase reorderings, and thus is unlikely to work well for the baseline system, but could work well for the systems where the source language has been reordered to mimic the target language. The translation systems were trained and tested using the Europarl corpus (Koehn, 2005). The training part contained 439513 sentences, where sentences longer than 40 words was filtered out. The test set has 2000 sentences and the development set had 500 sentences.

We performed two iterations of the iterative reordering rules learning and word alignment algorithm. After each iteration we trained a PBSMT system, which will be called Reo1 and Reo2, and which will be compared to baseline without any reorderings. Reo1 is similar to many previous approaches to reordering, since it is based on only one iteration of alignment and reordering.

5.1 Reordering Results

At each iteration, each training example was assigned to the training set with a probability of 0.8, and used for testing otherwise. Table 4 shows the number of examples of each type for the first two iterations. The data is rather skewed, with only around 10% of the examples being positive. Evaluating the rules on this automatically created test data gave a precision of around 55% and a recall of around 8% for the Swap class, in both iterations. Especially the recall is very low, but it can be compared to the recall of Elming (2008) of around 15%, which is also low. Table 3 shows a sample of the rules. Relatively few features are used in each rule, and it was quite common that not all of the four word sequences were used in the rules.

The number of rules was very different between the two iterations, with 77 rules in iteration 1 and only 14 rules in iteration 2. One possible explanation for this could be that the word alignments were improved, and thus that less rules that are due to noisy alignments were created, in iteration 2, but further investigation is needed to draw this conclusion. The function of the rules is also different between the iterations. Nearly all rules in iteration 2 concerns subject-verb inversion (d)[2]. The rules in iteration 1 are more varied, even though many move verbs towards the end of the sentence (a,b). Other examples of rules are those that handle adverb placement (c), but there are also some rules that are hard to explain linguistically, such as (e), which moves a noun to the end of a sentence. All linguitstic levels are used in the rules, and are often mixed. Out of the totally 91 rules, 28 are lexicalized (b,c). It is encouraging that new types of rules are learnt in iteration 2, but at the same time many of the useful rule types from iteration 1 unfortunately are missing.

5.2 Translation Results

Translation results are reported on the standard MT metrics Bleu (Papineni et al., 2002), Meteor (Lavie and Agarwal, 2007), and PER, position independent word error rate. PER does not take word order into account, which the other two metrics do.

The results with distortion penalty are presented in Table 5, and for monotone decoding in Table 6. As expected the results are overall higher with the distance-based reordering model in the decoder. On the systems with a distortion penalty there

[2] letters refer to rule ID in Table 3

System	Bleu	Meteor	1-PER
Base	20.15	26.87	0.712
Reo1	19.76	26.49	0.731
Reo2	20.13	26.99	0.736

Table 5: Results with distortion penalty

System	Bleu	Meteor	1-PER
Base	19.32	26.25	0.742
Reo1	19.40	26.39	0.737
Reo2	19.59	26.30	0.703

Table 6: Results with monotonous decoding

are very small differences between the systems on Bleu and Meteor, except for Reo1, which has the lowest score, whereas there is a small tendency of improvement for the systems with reordering on PER, which indicates that these systems are somewhat better with regard to lexical choice, which might be the result of better word alignment. For the monotone system there is a small tendency of improvement for the systems with reordering on Bleu and Meteor. The Reo2 system has a bad score on PER, however, indicating that this system likely has better word order than the other systems, since it has the highest Bleu score.

6 Conclusion

We have presentend a novel approach to reordering for SMT that could potentially improve both reordering rule learning and word alignment, by applying them iteratively. Initial experiments show that the rules we learn change with each iteration, to a large extent targeting different phenomena. The results on the SMT task, however, do not show any overall improvements; the systems with reordering largely perform on par with the baseline system without external reorderings. We still believe that the novel iterative approach can be useful, especially since we have shown that we learn different linguistically motivated rules in each iteration. Besides, there are plenty of room for improvements to the application of the main algorithm, such as using a different rule learning algorithm, preferably with a better accuracy of its rules, and by using a reordering lattice as translation input instead of 1-best input, which has been successful in previous research. We also want to investigate using a higher number of iterations, and of combining rules phrase tables and/or with high accuracy from different iterations.

References

William W. Cohen. 1995. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, USA.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd Annual Meeting of the ACL*, pages 531–540, Ann Arbor, Michigan, USA.

Jakob Elming. 2008. *Syntactic Reordering in Statistical Machine Translation*. Ph.D. thesis, Copenhagen Business School, Denmark.

Maria Holmqvist, Sara Stymne, Jody Foo, and Lars Ahrenberg. 2009. Improving alignment for SMT by reordering and augmenting the training corpus. In *Proceedings of the Fourth Workshop on Statistical Machine Translation*, pages 120–124, Athens, Greece.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondrej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the ACL, demonstration session*, pages 177–180, Prague, Czech Republic.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86, Phuket, Thailand.

Alon Lavie and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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 ACL*, pages 311–318, Philadelphia, Pennsylvania, USA.

Kay Rottmann and Stephan Vogel. 2007. Word reordering in statistical machine translation with a POS-based distortion model. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation*, pages 171–180, Skövde, Sweden.

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*, pages 508–514, Geneva, Switzerland.

ISSN 1736-6305 Vol. 11
http://hdl.handle.net/10062/16955