Dependency Forest based Word Alignment

Hitoshi Otsuki[1], **Chenhui Chu**[2], **Toshiaki Nakazawa**[2], **Sadao Kurohashi**[1]

[1]*Graduate School of Informatics, Kyoto University*
[2]*Japan Science and Technology Agency*

{otsuki, kuro}@nlp.ist.i.kyoto-u.ac.jp, {chu,nakazawa}@pa.jst.jp

Abstract

A hierarchical word alignment model that searches for k-best partial alignments on target constituent 1-best parse trees has been shown to outperform previous models. However, relying solely on 1-best parses trees might hinder the search for good alignments because 1-best trees are not necessarily the best for word alignment tasks in practice. This paper introduces a dependency forest based word alignment model, which utilizes target dependency forests in an attempt to minimize the impact on limitations attributable to 1-best parse trees. We present how k-best alignments are constructed over target-side dependency forests. Alignment experiments on the Japanese-English language pair show a relative error reduction of 4% of the alignment score compared to a model with 1-best parse trees.

1 Introduction

In statistical machine translation (SMT), word alignment plays an essential role in obtaining phrase tables (Och and Ney, 2004; Koehn et al., 2003) or syntactic transformation rules (Chiang, 2007; Shen et al., 2008). IBM models (Brown et al., 1993), which are based on word sequences, have been widely used for obtaining word alignments because they are fast and their implementation is available as GIZA++.[1]

Recently, a hierarchical alignment model (whose implementation is known as Nile[2]) (Riesa et al., 2011), which performs better than IBM models, has been proposed. In the hierarchical alignment model, both source and target constituency trees are used for incorporating syntactic information as features, and it searches for k-best partial alignments on the target constituent parse trees. It achieved significantly better results than the IBM Model4 in Arabic-English and Chinese-English word alignment tasks, even though the model was trained on only 2,280 and 1,102 parallel sentences as gold standard alignments. However, their models rely only on 1-best source and target side parse trees, which are not necessarily good for word alignment tasks.

In SMT, forest-based decoding has been proposed for both constituency and dependency parse trees (Mi et al., 2008; Tu et al., 2010). A forest is a compact representation of n-best parse trees. It provides more alternative parse trees to choose from during decoding, leading to significant improvements in translation quality. In this paper, we borrow this idea to build an alignment model using dependency forests rather than 1-best parses, which makes it possible to provide the model with more alternative parse trees that may be suitable for word alignment tasks. The motivation of using dependency forests instead of constituency forests in our model is that dependency forests are more appropriate for alignments between language pairs with long-distance reordering, such as the one we study in this paper. This is because they are more suitable for capturing the complex semantic relations of words in a sentence (Kahane, 2012).

We conducted alignment experiments on the Japanese-English language pair. Experimental results show a relative error reduction of 4% of the alignment score compared to the model with 1-best parse trees.

2 Model Description

2.1 Dependency Forest

We first briefly explain dependency forests that are used in our model before describing the alignment

[1]http://www.statmt.org/moses/giza/GIZA++.html
[2]http://jasonriesa.github.io/nile/

Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics – Student Research Workshop, pages 8–14,
Berlin, Germany, August 7-12, 2016.

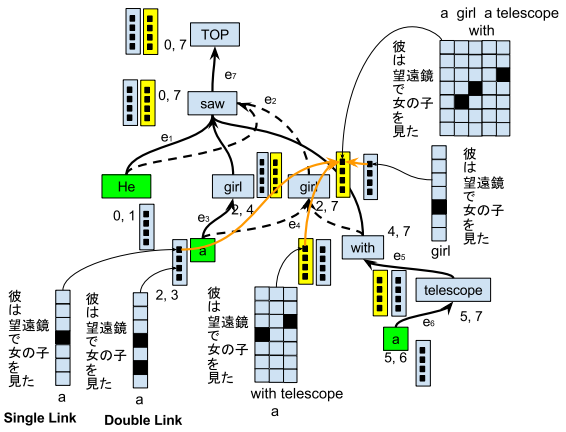


Figure 1: Bottom-up search for alignments over target-side dependency forest (This forest encodes 2-best parse trees for the sentence "he saw a girl with a telescope." The source sentence is "彼 (He) は望遠鏡 (telescope) で (with) 女の子 (girl) を見た (saw)". There are two interpretations for this sentence; either "with a telescope" depends on "saw" or "boy.")

Input : n-best dependency parses $\{T_i\}_{i=1}^{n}$ of a sentence
Score of T_i $Score_i$
Output: A forest F of $\{T_i\}_{i=1}^{n}$

1 F = `CreateForestStructure`$(\{T_i\}_{i=1}^{n})$
2 $edgeScores = \{\}$
3 $minScore = Min(\{Score_i\}_{i=1}^{n})$
4 **for** $i = 1$ **to** n **do**
5 $\quad Score_i -= minScore$
6 $\quad$ **for** $edge \in T_i$ **do**
7 $\quad\quad edgeScores\,[edge] += \frac{1}{n} Score_i$
8 $\quad$ **end**
9 **end**
10 **for** $hyperEdge \in F$ **do**
11 $\quad$ **for** $edge \in hyperEdge$ **do**
12 $\quad\quad hyperEdge.score += edgeScores\,[edge]$
13 $\quad$ **end**
14 **end**

Algorithm 1: Computation of a hyperedge score

construction method. A dependency forest is represented by a hypergraph $\langle V, E \rangle$, where V is a set of nodes and E is a set of hyperedges.

A hyperedge e connects nodes in the forest and is defined to be a triple $\langle tails(e), head(e), score \rangle$, where $tails(e)$ is a set of dependents of e, $head(e)$ is the head of e, and $score$ is the score of e that is usually obtained by heuristics (Tu et al., 2010). For example, e_1 in Figure 1 is equal to $\langle (he_{0,1}, boy_{2,4}, with_{4,7}), saw_{0,7}, 1.234 \rangle$. In our model, we use Algorithm 1 to compute hyperedge scores. Edges in a hyperedge are defined to be the ones obtained by connecting each tail with the head (Line 11). Hyperedge score is the sum of all the scores of edges in it (Line 12). The score of an edge is the normalized sum of the scores of all parses which contain the edge (Line 7).

Every node in a dependency forest corresponds to a word attached with a **span**, which is a range of word indices covered by the node. Following (Tu et al., 2010), a span is represented in the form i, j, which indicates the node covers all the words from i-th to $(j-1)$-th word. This requires dependency forests to be projective. Separate nodes are used for a word if the nodes in dependency trees have different spans. For example, in Figure 1 there are two nodes for the word "boy" because they have different spans (i.e., (2, 4) and (2, 7)).

The construction of a dependency forest from dependency trees is done by sharing the common nodes and edges (Line 1). The common nodes are those with the same span and part-of-speech (POS) . Note that the dependency forest obtained from this method does not necessarily encode exactly the dependency trees from which they are created. Usually there are more trees that can be extracted from the dependency forests (Boullier et al., 2009). In our experiment, when we use the term "a n-best dependency forest", we indicate a dependency forest that is created from n-best dependency trees.

2.2 Finding Alignments over Forest

Following the hierarchical alignment model (Riesa et al., 2011), our model searches for the best alignment by constructing partial alignments (hypotheses) over target dependency forests in a bottom-up manner as shown in Figure 1.

The algorithm for constructing alignments is shown in Algorithm 2. Note that source dependency forests are included in the input to the algorithm. This is optional but can be included for richer features. Each node in the forest has partial alignments sorted by alignment scores. Because it is computationally expensive to keep all possible partial alignments for each node, we keep a beam size of k. A partial alignment for a node is an alignment matrix for target words that are cov-

ered by the node. In Figure 1, each partial alignment is represented as a black square. Scores of the partial alignments are a linear combination of features. There are two types of features: local and non-local features. A feature f is defined to be local if and only if it can be factored among the local productions in a tree, and non-local otherwise (Huang, 2008).

We visit the nodes in the topological order, to guarantee that we visit a node after visiting all its tail nodes (Line 1). For each node, we first generates partial alignments, which are one column alignment matrices for its word. Because of time complexity, we only generates null, single link and double link alignment (Line 5). A single and double link alignment refer to a column matrix having exactly one and two alignments, respectively, as shown in Figure 1. For each partial alignment, we compute its score using local features (Line 7) and pushed to a priority queue B_v (Line 8). These partial alignments are represented by black squares in a blue container in Figure 1. Then, we compute partial alignments for the target words covered by the node, by combining tails' partial alignments and one column alignments for its word using non-local features (Line 10 - 14), which is represented by the orange arrows in Figure 1. k-best combined partial alignments are put in Y_v (Line 14). They are represented by black squares in a yellow container in Figure 1. Here, we use cube pruning (Chiang, 2007) to get the approximate k-best combinations. Note that in the search over constituency parse trees, one column alignment matrices are generated only on the leaf node (Riesa et al., 2011), whereas we generate them also on non-leaf nodes in the search over dependency forests.

Input : Source and target sentence $\mathbf{s}, \mathbf{t}$
Dependency forest F_s over $\mathbf{s}$
Dependency forest F_t over $\mathbf{t}$
Set of feature functions $\mathbf{h}$
Weight vector $\mathbf{w}$
Beam size k
Output: A k-best list of alignments over $\mathbf{s}$ and $\mathbf{t}$

1 **for** $v \in$`TopologicalSort(`F_t`)` **do**
2 $links = \emptyset$
3 $B_v = \emptyset$
4 $i =$ `word-index-of(`v`)`
5 $links = \{(0, i)\}$ `USingleLinks(`i`)` `UDoubleLinks(`i`)`
6 **for** $link \in links$ **do**
7 $score = \mathbf{w} \cdot \mathbf{h}(links, v, \mathbf{s}, \mathbf{t}, F_s, F_t)$
8 `Push(`B_v, $\langle score, link \rangle$, k`)`
9 **end**
10 **for** $hyperEdge \in$`InHyperEdges(`v`)` **do**
11 $c = hyperEdge.tail$
12 `Push(`α_v, $\langle Y_{c_1}, \cdots, Y_{c_{|c|}}, B_v \rangle$`)`
13 **end**
14 $Y_v =$`CubePruning(`α_v, k, $\mathbf{w}$, $\mathbf{h}$, v, $\mathbf{s}$, $\mathbf{t}$, F_s, F_t`)`
15 **end**

Algorithm 2: Construction of alignments

2.3 Features

The features we used include those used in Nile except for the automatically extracted rule and constellation features. This is because these features are not easily applicable to dependency forests. As shown in our experiments, these features have a contribution to the alignment score. However, our primary purpose is to show the effect of using forests on alignment quality.

Several features in Nile such as source-target POS local feature and coordination feature have to be customized for dependency forests, because it is possible that there are multiple nodes that correspond to the same word. We decided to consider all nodes corresponding to a word by counting the frequency of each POS tag of a node, and normalizing it with the total frequency of POS tags in the forest. For example, suppose there are four nodes which correspond to the same word, whose POS tags are JJ, VBG, JJ, VGZ. In this case the features "src-tgt-pos-feature-JJ=0.5", "src-tgt-pos-feature-VBG=0.25" and "src-tgt-pos-feature-VBZ=0.25" are activated.

Besides the features used in Nile, our model uses a contiguous alignment local feature and a hyperedge score non-local feature. The contiguous alignment feature fires when a target word is aligned to multiple source words, and these words are contiguous on a forest. Preliminary experiments showed, however, that none of these features contributed to the improvement of the alignment score.

3 Experiments

3.1 Experimental Settings

We conducted alignment experiments on the Japanese-English language pair. For dependency

parsers, we used KNP (Kawahara and Kurohashi, 2006) for Japanese and Berkeley Parser (Petrov and Klein, 2007) for English. We converted constituent parse trees obtained by Berkeley Parser to dependency parse trees using rules.[3] We used 300, 100, 100 sentences from ASPEC-JE[2] for training, development and test data, respectively.[4] Our model as well as Nile has a feature called third party alignment feature, which activates for an alignment link that is presented in the alignment of a third party model. The beam size k was set to 128. We used different number of parse trees to create a target forest, e.g., 1, 10, 20, 50, 100 and 200.[5] The baseline in this experiment is a model with 1-best parse trees on the target side. For reference, we also experimented on Nile[6], the Bayesian subtree alignment model (Nakazawa model) (Nakazawa and Kurohashi, 2011) and IBM Model4.[7] We used Nile without automatically extracted rule features and constellation features to make a fair comparison with our model.

3.2 Results

Table 1 shows the alignment results evaluated on precision, recall and F-score for each experimental setting. The first row shows the names of different experimental settings. Each number in the row shows the number of n-best parse trees used to create target forests.

We can observe that using forests improves the score. However, the improvement does not monotonically increase with the number of trees on the target side. When 100-best is used in target side, it achieved the highest error reduction of 4% compared to the baseline model. [8]

We also conducted experiments on different number of beam size k, e.g, 200 and 300, from the insight that a larger number of trees encoded in a forest indicates that more noisy partial alignments are generated, using the same k as the 1-best model is not sufficient. However, we could not observe significant improvements.

[3]The conversion program is available at https://github.com/hitochan777/mt-tools/releases/tag/1.0.1

[4]http://lotus.kuee.kyoto-u.ac.jp/ASPEC/

[5]In the experiments, we used 1-best parse trees for the source side. Although our model also allows to use forests on the source side, preliminary experiments showed that using forests on the source side does not improve the alignment score.

[6]Note that Nile uses 1-best constituency parse tree

[7]The alignments from Nakazawa model and IBM Model 4 were symmetrized with the grow-diag-final heuristic.

[8]$(82.39 - 81.66) \,/\, (100 - 81.66) \approx 4\%$

4 Discussion

We observed the improvement of alignments by using forests. We checked whether good parse trees were chosen when higher F-scores were achieved. It turned out that better parse trees led to higher F-scores, as shown in Figure 2a, but it was not always the case.

Figure 2a shows an improved example by using 100-best trees on the target side. In the figure, we can observe that "$\mathcal{O}$" and "of" are correctly aligned. We observe that the English 1-best parse tree is incorrect, whereas 100-best model were able to choose a better tree.

Figure 2b shows a worsened example by using 200-best trees on the target side. We can see that the 200-best model aligned many words unnecessarily and the wrong tree is chosen even though the 1-best parse is good. There were many cases in which forests are harmful for alignments. There are two possible reasons. Firstly, most of the features in our model comes from Nile, but they are not informative enough to choose better parses from forests. Secondly, our model is likely to suffer from the data sparseness because using forests generates more noise than 1-best parses.

For our model to benefit from forests we have to consider the following: Firstly, our model's feature is based on the assumption that source and target forests contain trees with similar structures to each other. However the projectivity of forests prohibits our model from generating (choosing) target trees that are similar to the ones in source forests. Secondly, we observed the cases where no parse in forests captures the correct root and the difference of n-best parses are mainly POS tags of words.

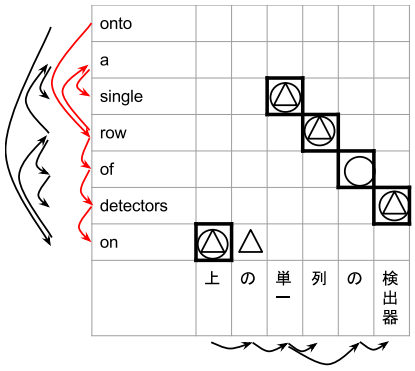
Our model performs on par with Nile because our model is based on Nile. However, our model outperforms the Nakazawa model and IBM Model4. This is because our model is supervised but these models are unsupervised. The Nakazawa model outperformed IBM Model4 because it utilizes dependency trees, which provide richer information.

5 Related Work

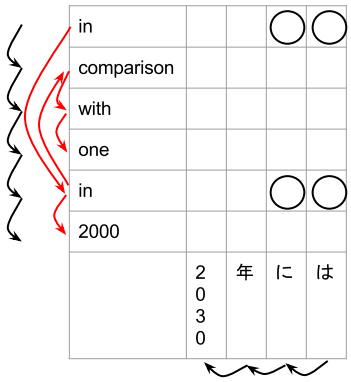
Studies have been conducted to make use of more alternatives to cope with the unreliability of 1-best results. Liu et al. (2009) proposed a structure called weighted alignment matrix, which encodes the distribution over all possible alignments.

Model	1	10	20	50	100	150	200	Nile	Nakazawa	IBM Model 4
Precision	82.56	82.90	83.51	83.28	**83.77**	83.34	83.39	83.26	70.59	63.21
Recall	80.79	80.88	80.62	80.75	81.05	80.66	80.75	81.52	**82.67**	74.25
F-score	81.66	81.87	82.04	81.99	**82.39**	81.98	82.05	82.38	76.16	68.29

Table 1: Precision, Recall and F-score for ASPEC-JE. The numbers in the first row refer to the number of k-best parse trees used to generate forests.



(a) 1-best and 100-best model comparison in the alignment of "onto a single row of detectors on" and "上 (on) の (of) 単一 (single) 列 (row) の (of) 検出器 (detectors)"



(b) 1-best and 200-best model comparison in the alignment of "in comparison with one in 2000" and "2030(2030) 年 (year) には (in)"

Figure 2: Alignment result: Black boxes represent golden alignments. Triangles represent 1-best model alignments. Circles represent the alignments of proposed model. Black and red arcs represent 1-best parses and chosen parses respectively.

They introduced a way to extract phrase pairs and estimate their probabilities. Their proposed method outperformed the baseline which uses n-best alignments. Venugopal et al. (2008) used n-best alignments and parses to generate fraction counts used for machine translation downstream estimation. While their approaches are to use n-best alignments already obtained from some alignment models, our model finds k-best list of alignments for given sentences.

Mi et al. (2008) and Tu et al. (2010) used packed constituency forests and dependency forests respectively for decoding. The best path that is suitable for translation is chosen from the forest during decoding, leading to significant improvement in translation quality. Note that they do not use forests for obtaining word alignments.

The approaches for modeling word alignment can be divided into two categories: discriminative models (Dyer et al., 2011; Setiawan et al., 2010) and generative models (Brown et al., 1993; Nakazawa and Kurohashi, 2011). Generative models such as the IBM models (Brown et al., 1993) have the advantage that they do not require golden alignment training data annotated by humans. However, it is difficult to incorporate arbitrary features in these models. On the other hand, discriminative models can incorporate arbitrary features such as syntactic information, but they generally require gold training data, which is hard to obtain in large scale. For discriminative models, word alignment models using deep neural network have been proposed recently (Tamura et al., 2014; Songyot and Chiang, 2014; Yang et al., 2013).

6 Conclusion

In this work, we proposed a hierarchical alignment model based on dependency forests, which advanced an alignment model that uses constituency parse trees (Riesa et al., 2011) to allow to use more suitable parse trees for word alignment. Experimental results on the Japanese-English language pair show a relative error reduction of 4% of the alignment score compared to a model with 1-best parse trees that using forest on the target side.

Our future work will involve the implementation of missing features, because the automatic translation rule features had a large contribution to the improvement of alignment quality in Nile.

The experimental results show that Nile, which uses 1-best constituency parses , had almost the same F-score as our proposed method with 100-best parse trees. It will be interesting to see the effect of using forests in Nile.

Moreover, we are considering to investigate the efficacy of our model with different parsers and language pairs.

Finally, we are also considering using training data with richer information such as the one described in (Li et al., 2010).

Acknowledgements

We thank Graham Neubig for his valuable comments during a pre-submission mentoring program, which had greatly improved the quality of the manuscript. We also thank the anonymous reviewers for their helpful comments.

References

Pierre Boullier, Alexis Nasr, and Benoît Sagot. 2009. Constructing parse forests that include exactly the n-best pcfg trees. In *Proceedings of the 11th International Conference on Parsing Technologies (IWPT'09)*, pages 117–128, Paris, France, October. Association for Computational Linguistics.

Peter F Brown, Vincent J Della Pietra, Stephen A Della Pietra, and Robert L Mercer. 1993. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311.

David Chiang. 2007. Hierarchical phrase-based translation. *Computational Linguistics*, 33(2):201–228.

Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. 2011. Unsupervised word alignment with arbitrary features. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 409–419. Association for Computational Linguistics.

Liang Huang. 2008. Forest reranking: Discriminative parsing with non-local features. In *Association for Computational Linguistics*, pages 586–594.

Sylvain Kahane. 2012. Why to choose dependency rather than constituency for syntax: a formal point of view. *J. Apresjan, M.-C. L' Homme, M.-C. Iomdin, J. Milicevic, A. Polguère, and L. Wanner, editors, Meanings, Texts, and other exciting things: A Festschrift to Commemorate the 80th Anniversary of Professor Igor A. Mel' cuk*, pages 257–272.

Daisuke Kawahara and Sadao Kurohashi. 2006. A fully-lexicalized probabilistic model for japanese syntactic and case structure analysis. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, New York City, USA, June. Association for Computational Linguistics.

Philipp Koehn, Franz Josef Och, and Daniel Marcu. 2003. Statistical phrase-based translation. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 48–54. Association for Computational Linguistics.

Xuansong Li, Niyu Ge, Stephen Grimes, Stephanie Strassel, and Kazuaki Maeda. 2010. Enriching word alignment with linguistic tags. In *Language Resources and Evaluation Conference*.

Yang Liu, Tian Xia, Xinyan Xiao, and Qun Liu. 2009. Weighted alignment matrices for statistical machine translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 1017–1026. Association for Computational Linguistics.

Haitao Mi, Liang Huang, and Qun Liu. 2008. Forest-based translation. In *Association for Computational Linguistics*, pages 192–199.

Toshiaki Nakazawa and Sadao Kurohashi. 2011. Bayesian subtree alignment model based on dependency trees. In *International Joint Conference on Natural Language Processing*, pages 794–802.

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

Slav Petrov and Dan Klein. 2007. Improved inference for unlexicalized parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 404–411, Rochester, New York, April. Association for Computational Linguistics.

Jason Riesa, Ann Irvine, and Daniel Marcu. 2011. Feature-rich language-independent syntax-based alignment for statistical machine translation. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 497–507. Association for Computational Linguistics.

Hendra Setiawan, Chris Dyer, and Philip Resnik. 2010. Discriminative word alignment with a function word reordering model. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 534–544. Association for Computational Linguistics.

Libin Shen, Jinxi Xu, and Ralph Weischedel. 2008. A new string-to-dependency machine translation algorithm with a target dependency language model. In *Proceedings of ACL-08: HLT*, pages 577–585, Columbus, Ohio, June. Association for Computational Linguistics.

Theerawat Songyot and David Chiang. 2014. Improving word alignment using word similarity. In *Empirical Methods in Natural Language Processing*, pages 1840–1845. Citeseer.

Akihiro Tamura, Taro Watanabe, and Eiichiro Sumita. 2014. Recurrent neural networks for word alignment model. In *Antenna Measurement Techniques Association*, pages 1470–1480.

Zhaopeng Tu, Yang Liu, Young-Sook Hwang, Qun Liu, and Shouxun Lin. 2010. Dependency forest for statistical machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 1092–1100. Association for Computational Linguistics.

Ashish Venugopal, Andreas Zollmann, Noah A Smith, and Stephan Vogel. 2008. Wider pipelines: N-best alignments and parses in mt training. In *Proceedings of Antenna Measurement Techniques Association*, pages 192–201. Citeseer.

Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *Antenna Measurement Techniques Association*, pages 166–175.