# Hierarchical Word Alignment based on Dependency Forest

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@nlp.ist.i.kyoto-u.ac.jp, (chu,nakazawa)@pa.jst.jp, kuro@i.kyoto-u.ac.jp

## Abstract

This paper introduces dependency forest based word alignment model which utilizes both source and target dependency forests in an attempt to minimize the impact on parse errors in 1-best parse trees. Hierarchical word alignment model which 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 parse tree might hinder the search for good alignments because 1-best trees are not necessarily the best in practice. This paper describes how $k$-best alignments are constructed over target-side dependency forest. Alignment experiments on Japanese-English and Japanese-Chinese show that both the variation and structural similarity of source and target forest are important for our model to fully benefit from forests.

## 1 Introduction

In statistical machine translation (SMT), word alignment plays an essential role in obtaining phrase tables and syntactic transformation rules. The main approaches for modeling word alignment are by using discriminative models [2, 10] and generative models [1, 7]. Generative models such as the IBM models [1] 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.

There exist discriminative models which perform better than IBM models with relatively small training data such as the hierarchical alignment models, in which source and target constituency trees are used for incorporating syntactic information as features (whose implementation is known as Nile[1]) [9]. They achieve significantly better result than the IBM Model4 in Arabic-English and Chinese-English word alignment task even though their model was trained using only 2,280 and 1,102 parallel sentences as gold standard alignments.

However, these models are sensitive to parsing errors since they relies heavily on source and target trees. To alleviate parsing errors, we propose to use forests which are compact representation of $n$-best parses.

In SMT, forest-based translation has been proposed for both constituency and dependency parse trees [6, 13], which provides more alternative parse trees to choose from, and they lead to significant improvement in translation quality. We apply this idea to build an alignment model using dependency forests rather than 1-best parses, which provide more robustness against parsing errors. The motivation of using dependency trees is that they are more suitable for capturing the semantic relations of words regardless of their positions in a sentence.

We conducted alignment experiments on two language pairs; Japanese-English and Japanese-Chinese. Experimental results show that the structural similarity and the variation of source and target forest has a big impact on the alignment quality.

## 2 Model Description

### 2.1 Finding $k$-best alignments over forest

Following the hierarchical alignment model [9], our model searches for the best alignment by constructing partial alignments (hypotheses) over dependency forests for the target language in a bottom-up manner as shown in Figure 1. 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 if otherwise [3]. For a detailed explanation of the search algorithm, refer to [9].

A forest is a hypergraph $\langle V, E \rangle$ where $V$ is a set of nodes and $E$ is a set of hyperedges. A hyperedge $e$ is defined to be a pair $\langle tails(e), head(e), score \rangle$ where

[1] https://github.com/jasonriesa/nile

$tails(e)$ is a set of tails of $e$, $head(e)$ is a head of $e$, and *score* is a score of $e$ which is usually obtained by heuristics [13]. Each node has *span* of the form $i, j$ which means the node covers $i$-th to $(j-1)$-th node. An example of a forest is shown in Figure 1, which encodes 2-best parse trees.

We visit the nodes in the topological order, which guarantees that we visit a non-leaf node after computing $k$-best hypotheses of its children. The key difference between search over constituency tree and dependency forest is that every node in a dependency forest corresponds to a word whereas only leaf nodes correspond to words in a constituency tree. This requires us to compute $k$-best hypotheses for the word of each non-leaf node before combining the hypotheses from its children. Therefore, the hypotheses can be seen as the ones obtained in its virtual child which is a leaf node. We call this node "dummy child". In Figure 1, $k$-best hypotheses for a word are represented by a blue cylinder and each black square is a hypothesis. After computing $k$-best hypotheses for a node's children including its dummy child, cube pruning is applied to approximately find $k$-best hypotheses for the node which is represented by a yellow cylinder in Figure 1.

Notice that in many cases there are more than two children for a non-leaf node in dependency forests. Since we apply cube pruning to all the children at once, it is likely that only the highly-ranked hypotheses of each child are considered for the generation of $k$-best hypotheses for the parent node. This leads to problems when the highly-ranked hypotheses are in fact bad alignments from the global perspective. To solve this problem, we put all the children's hypotheses into a queue and repeatedly dequeue the first two elements, apply cube pruning and enqueue the obtained $k$-best hypotheses till there remains only one $k$-best hypotheses list in the queue. We call this method "binarized cube pruning".

## 2.2 Features

The features we used include those used in Nile except for the automatic rule extraction features and constellation features. This is because these features are not easily applicable to dependency forests.

Several features in Nile such as source-target part-of-speech (POS) local feature and coordination feature have to be customized for dependency forests since it is possible that there are multiple nodes which 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 corresponding to a target word 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.

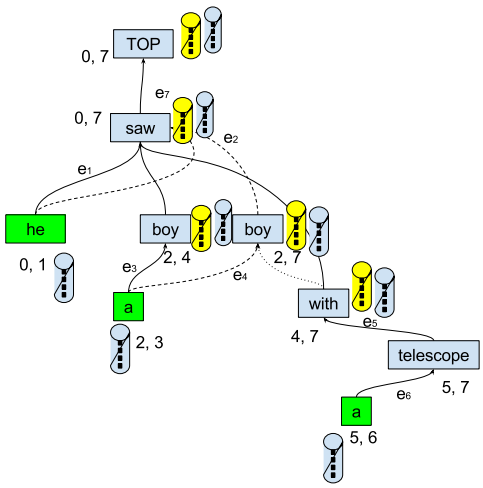


Figure 1: Bottom-up search for $k$-best alignments over dependency forest on target side

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

# 3 Experiments

## 3.1 Experimental Setting

We conducted alignment experiments on two language pairs, Japanese-English and Japanese-Chinese. For dependency parsers, we used KNP [4] for Japanese, berkeley Parser [8][2] and SKP [11] for English and SKP for Chinese. For Japanese-English, we used 300, 100, 100 sentences from ASPEC-JE for training, development and test data, respectively. For Japanese-Chinese, we used 310, 100, 100 sentences from ASPEC-JC for training, development and test data, respectively.[3] Our model as well as Nile has a feature called third party alignment feature which activates for a alignment link which is presented in the alignment of a third party model. In our experiment, we used two alignment models, IBM Model4 trained with GIZA++ and bayesian subtree alignment model based on dependency trees [7], both of which are symmetrized with the grow-diag-final heuristic. Moreover, in order to investigate how much the structural similarity of a source and a target parse tree affects the alignment quality,

[2]We converted constituent parse trees obtained by berkeley Parser to dependency parse trees using rule.

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

we also conducted an experiment using target forests obtained by projecting source trees to target trees by using the algorithm explained in [12] and encoding the projected trees into a forest.

### 3.2 Experimental Result

Figure 2, 3 shows precision, recall and F-score for each experimental setting. In these figures, $n_s(P_s)-n_t(P_t)$ implies that the forest of $n_s$-best trees parsed by a parser $P_s$ is used in the source side, and the forest of $n_t$-best trees parsed by a parser $P_t$ is used in the target side. *bin* means binarized cube pruning is used. *proj* means a forest of projected target trees are used. We use Nile as the baseline system. We were unable to implement two features (e.g. the automatic rule extraction features and constellation features) as they are not easily applicable to dependency forests. These features are turned off in Nile to make a fair comparison with our model, which is denoted as *Nile-*. *Nile* uses all features. *Nakazawa* is a Bayesian subtree alignment model based on dependency trees [7] Finally, GIZA++ is the IBM Model4. The last three are shown only for reference.

For Japanese-Chinese, negative effect on the alignment quality is observed by using forests. Note that, the recall tends to drop with the growth in the size of target side forest. Binarized cube pruning does not have a positive effect in ASPEC-JC. However we can see the improvement of alignment quality by using projected trees on the target side. When 1-best projected trees are used on the target side, F-score is even higher than the baseline (Nile with all features). But unfortunately using more projected trees does not help to further improve the score. It is also worth noticing that the results of using 10-best projected trees and 20-best projected trees on the target side are almost the same.

For Japanese-English, we can see that using forests improves the score but the improvement does not monotonically increase with the number of trees on the target side. Unlike ASPEC-JC, we can see a relatively big improvement in F-score when binarized cube pruning is used. Like ASPEC-JC, we can see an improvement by using projected trees on the target side, but using more trees did not improve the score. Note that, even if we use projected trees on the target side, the score is far behind that of Nile with all features.

## 4 Discussion

As can be seen in the experimental results, the improvement in F-score by using projection of source tree to target tree shows that the structural similarity of source and target tree is important. Our model's performance will improve if we can use higher quality parses of source and target side sentences and exploit structural similarity in a better way.

We observed that F-scores obtained when 10-best and 20-best projected trees are used had no noticeable difference. To understand this phenomenon we checked the variation of source trees from which we project to target trees. It turns out that the average number of trees for a sentence in ASPEC-JC was 4.30 for 10-best and 4.97 for 20-best. This indicates that KNP outputs limited number of trees. Therefore, we believe that our model can perform better if there is more variation in the source side trees when the projection is applied.

For Japanese-English, we observed the improvement of alignments by using forests. We checked whether good parse trees were chosen when higher F-scores were achieved. However, it turned out that higher F-scores, in most cases, were not results of better parse trees. This might imply that good parse trees are not always needed for better word alignment.

## 5 Conclusion

In this work, we proposed a hierarchical alignment model with dependency forests based on the alignment model which uses constituency parse trees [9] to address parse errors. Experimental results show that the structural similarity of source and target forest has a big impact on the alignment quality. Our future work will involve the implementation of missing features because the automatic translation features had a large contribution to the improvement of alignment quality in Nile. Also, we need to figure out some methods to choose trees with good quality and high structural similarity to fully benefit from forests. Finally, we are also considering using training data with richer information such as the one described in [5].

## References


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

[2] Chris Dyer, Jonathan Clark, Alon Lavie, and Noah A Smith. 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, 2011.

[3] Liang Huang. Forest reranking: Discriminative parsing with non-local features. In *ACL*, pages 586–594, 2008.

[4] Daisuke Kawahara and Sadao Kurohashi. A fully-lexicalized probabilistic model for japanese syntactic and case structure analysis. In *Proceedings of*

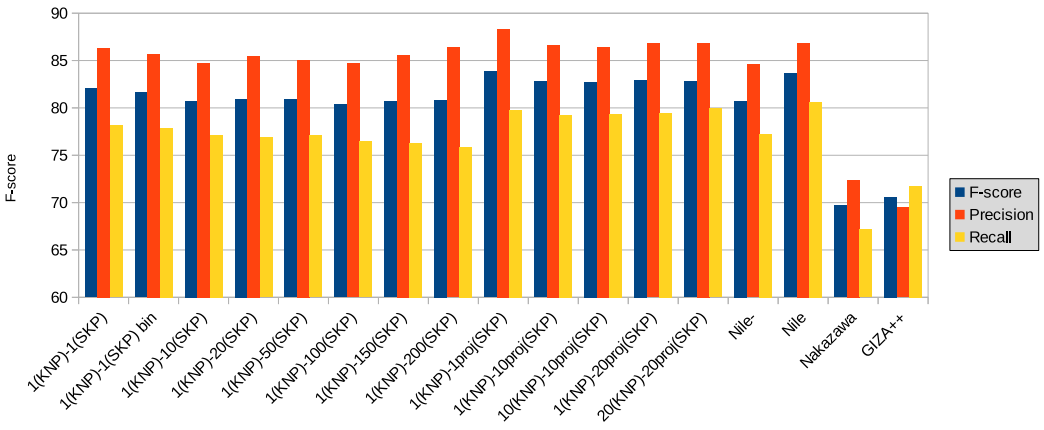


Figure 2: Precision, Recall and F-score for ASPEC-JC

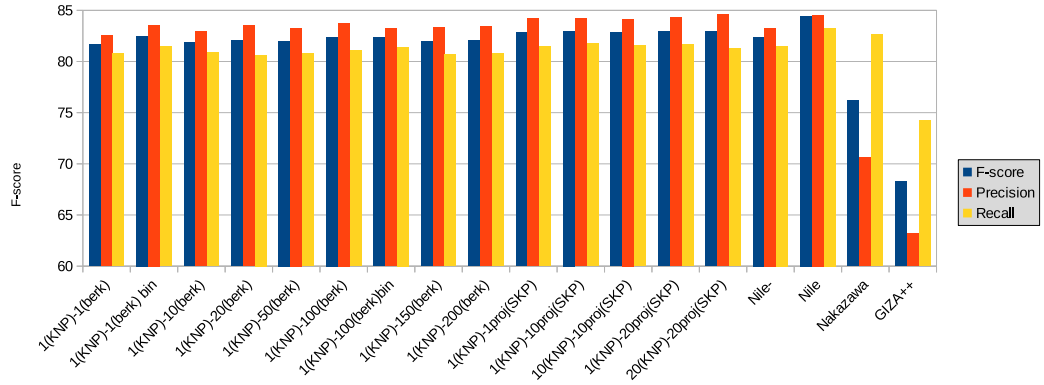


Figure 3: Precision, Recall and F-score for ASPEC-JE

*the Human Language Technology Conference of the NAACL, Main Conference*, pages 176–183, New York City, USA, June 2006. Association for Computational Linguistics.

[5] Xuansong Li, Niyu Ge, Stephen Grimes, Stephanie Strassel, and Kazuaki Maeda. Enriching word alignment with linguistic tags. In *LREC*, 2010.

[6] Haitao Mi, Liang Huang, and Qun Liu. Forest-based translation. In *ACL*, pages 192–199, 2008.

[7] Toshiaki Nakazawa and Sadao Kurohashi. Bayesian subtree alignment model based on dependency trees. In *IJCNLP*, pages 794–802, 2011.

[8] Slav Petrov and Dan Klein. 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 2007. Association for Computational Linguistics.

[9] Jason Riesa, Ann Irvine, and Daniel Marcu. 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, 2011.

[10] Hendra Setiawan, Chris Dyer, and Philip Resnik. 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, 2010.

[11] Mo Shen, Daisuke Kawahara, and Sadao Kurohashi. A reranking approach for dependency parsing with variable-sized subtree features. In *Proceedings of the 26th Pacific Asia Conference on Language, Information, and Computation*, pages 308–317, Bali,Indonesia, November 2012. Faculty of Computer Science, Universitas Indonesia.

[12] Yu Shen, Chenhui Chu, Fabien Cromieres, and Sadao Kurohashi. Cross-language projection of dependency trees for tree-to-tree machine translation. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computing*, pages 80–88, 2015.

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