A Simple, Fast Strategy for Weighted Alignment Hypergraph

Zhaopeng Tu[1,2,⋆], Jun Xie[2], Yajuan Lv[2], and Qun Liu[2,3]

[1] Department of Computer Science,
University of California, Davis, USA
zptu@ucdavis.edu
[2] Key Laboratory of Intelligent Information Processing,
Institute of Computing Technology, CAS, Beijing, China
{tuzhaopeng,xiejun,lvyajuan,liuqun}@ict.ac.cn
[3] Centre for Next Generation Locolisation
Dublin City University, Ireland
qliu@computing.dcu.ie

Abstract. Weighted alignment hypergraph [4] is potentially useful for statistical machine translation, because it is the first study to simultaneously exploit the compact representation and fertility model of word alignment. Since estimating the probabilities of rules extracted from hypergraphs is an NP-complete problem, they propose a divide-and-conquer strategy by decomposing a hypergraph into a set of independent subhypergraphs. However, they employ a Bull's algorithm to enumerate all consistent alignments for each rule in each subhypergraph, which is very time-consuming especially for the rules that contain non-terminals. This limits the applicability of this method to the syntax translation models, the rules of which contain many non-terminals (e.g. SCFG rules). In response to this problem, we propose an inside-outside algorithm to efficiently enumerate the consistent alignments. Experimental results show that our method is twice as fast as the Bull's algorithm. In addition, the efficient dynamic programming algorithm makes our approach applicable to syntax-based translation models.

Keywords: statistical machine translation, weighted alignment hypergraph, optimization.

1 Introduction

Word alignment is the task of identifying translational relations (alignment links) between words in parallel corpora, in which a word at one language is usually translated into several words at the other language [1]. Following Moore [6], we divide alignment links into four categories: "one-to-one" (1-1), "one-to-many" (1-n), "many-to-one" (m-1) and "many-to-many" (m-n). Table 1 shows the distribution of links in a word-aligned corpus that contains 1.5 million sentence

⋆ Corresponding author.

G. Zhou et al. (Eds.): NLPCC 2013, CCIS 400, pp. 188–199, 2013.

pairs. From this table, we can see that nearly half of the links are not 1-1 links in both 1-best alignments and 10-best lists. This analysis suggests that the links are not irrelevant to each other and it is necessary to pay attention to the relations among them.

Table 1. The distribution of alignment links in a word-aligned corpus that contains 1.5 million sentence pairs

Alignments	1-1	1-n	m-1	m-n
1-best	56.5%	17.4%	12.9%	13.2%
10-best	56.4%	17.3%	12.8%	13.5%

To model this phenomenon, Liu et al., [4] propose a novel graph-based compact representation of word alignment, which takes into account the joint distribution of alignment links. They first transform each alignment to a bipartite graph (*bigraph* in short), in which the nodes are the words in the source and target sentences, and the edges are word-by-word links. Each bigraph can be decomposed into a set of disjoint minimal subgraphs, each of which is connected and corresponds to a set of interrelated links. These subgraphs work as fundamental units in the proposed approach to exploit the relations among the links. Then they employ a weighted alignment hypergraph to encode multiple bigraphs, in which each hyperedges corresponds to a subgraph in the bigraphs (§ 2.1).

Since estimating the probabilities of rules extracted from hypergraphs is an NP-complete problem, they propose a divide-and-conquer strategy by decomposing a hypergraph into a set of independent subhypergraphs. Indeed, the reduced number of hyperedges in the subhypergraphs make the strategy computationally tractable (§ 2.2). However, they employ a Bull's algorithm to enumerate all consistent alignments for each rule in each subhypergraph, which is very time-consuming especially for the rules that contain non-terminals (§ 2.3). This limits the applicability of this method to the syntax translation models, the rules of which contain many non-terminals (e.g. SCFG rules).

To alleviate this problem, we propose an inside-outside algorithm to further divide the subhypergraph into two independent parts (§ 3). With the inside-outside algorithm, we can employ shared structures for efficient dynamic programming. This is very important when calculating the probabilities of rules with non-terminals. Experimental results show that our approach is twice as fast as the conventional Bull's algorithm (§ 4.2). Specifically, the fact that we spend much less time in the rules with non-terminals makes our approach applicable to syntax-based translation models, whose rules contain many non-terminals (e.g. SCFG rules) (§ 4.2).

2 Background

2.1 Weighted Alignment Hypergraph

Each alignment of a sentence pair can be transformed to a bigraph, in which the two disjoint vertex sets S and T are the source and target words respectively, and the

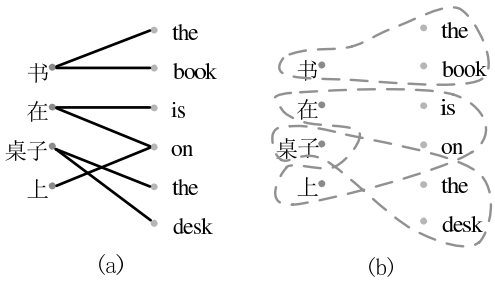

Fig. 1. (a) An example of bigraph constructed from an alignment between a pair of Chinese and English sentences, (b) the disjoint subgraphs of the bigraph in (a)

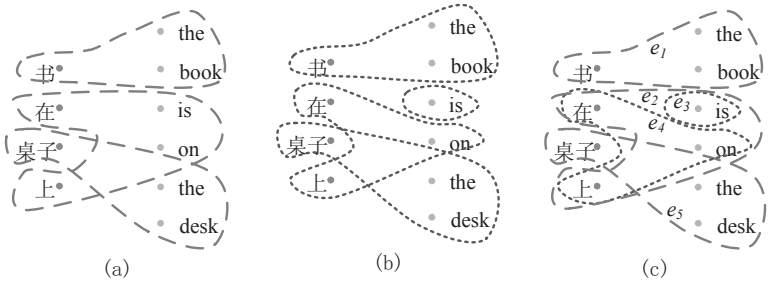

Fig. 2. (a) One alignment of a sentence pair; (b) another alignment of the same sentence pair; (c) the resulting hypergraph that takes the two alignments as samples

edges are word-by-word links. For example, Figure 1(a) shows the corresponding bigraph of an alignment. Since the bigraph usually is not connected, it can be decomposed into a unique set of *minimum connected subgraphs* (MCSs), where each subgraph is connected and does not contain any other MCSs. For example, the bigraph in Figure 1(a) can be decomposed into the MCSs in Figure 1(b). We can see that each MCS corresponds to a many-to-many link. Hereinafter, we use a bigraph to denote an alignment of a sentence pair.

A *weighted alignment hypergraph* [4] is a hypergraph that compactly encodes multiple bigraphs. For example, Figures 2(a) and 2(b) show two bigraphs of the same sentence pair. Then, a weighted alignment hypergraph can be constructed by encoding the union set of MCSs in each bipartite hypergraph, in which each MCS serves as a hyperedge, as in Figure 2(c). Accordingly, each hyperedge is associated with a weight to indicate how well it is, which is the probability sum of bigraphs in which the corresponding MCS occurs divided by the probability sum of all possible bigraphs.

Formally, a *weighted bipartite hypergraph* H is a triple $\langle S, T, E \rangle$ where S and T are two sets of vertices on the source and target sides, and E are hyperedges associated with weights. Liu et al., [4] estimate the weights of hyperedges from an n-best list by calculating relative frequencies:

$$w(e_i) = \frac{\sum_{BG \in \mathcal{N}} p(BG) \times \delta(BG, g_i)}{\sum_{BG \in \mathcal{N}} p(BG)}$$

where

$$\delta(BG, g_i) = \begin{cases} 1 & g_i \in BG \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Here $\mathcal{N}$ is an n-best bigraph (i.e., alignment) list, $p(BG)$ is the probability of a bigraph BG in the n-best list, g_i is the MCS that corresponds to e_i, and $\delta(BG, g_i)$ indicates that whether a subgraph g_i occurs in the bigraph BG or not.

2.2 Calculating Rule Probabilities

Liu et al., [4] calculate the fractional count of a phrase pair extracted from the hypergraph as the probability sum of the alignments with which the phrase pair is consistent, divided by the probability sum of all alignments encoded in a hypergraph. Therefore, they need to calculate two probability sums:

1. How to calculate the probability sum of all alignments encoded in a hypergraph?
2. How to efficiently calculate the probability sum of all consistent alignments for each phrase pair?

Enumerating All Alignments
Liu et al., [4] show that enumerating all possible alignments in a hypergraph can be reformulated as finding all possible *complete hypergraph matchings* in this

Algorithm 1. Algorithm for enumerating all possible complete hyperedge matchings in a bipartite hyperedge $H = \langle S, T, E \rangle$.

```
1: procedure ENUMERATION(⟨S, T, E⟩)
2:     completes ← ∅
3:     paths ← {∅}
4:     for e in E do
5:         new_paths ← ∅
6:         for path in paths do
7:             if e ∩ path == ∅ then
8:                 new_path ← path ∪ {e}
9:                 if new_path connects all vertices then
10:                    add new_path to completes
11:                else
12:                    add new_path to new_paths
13:        add new_paths to paths
14:    return completes
```

hypergraph, an NP-complete problem. Therefore, they propose a divide-and-conquer strategy by decomposing a hypergraph into a set of independent subhypergraphs. For example, Figure 3(b) shows the independent subhypergraphs of the hypergraph in Figure 3(a). The reduced number of hyperedges in the subhypergraphs makes the strategy computationally tractable. For each subhypergraph, they employ a Bull's algorithm for enumerating all possible complete hyperedge matchings in each subhypergraph, as shown in Algorithm 1. The complexity of this algorithm is $O(|E|)$, where $|E|$ is the number of hyperedges in the hypergraph.

Enumerating Consistent Alignments
To efficiently calculate the probability sum of all consistent alignments for each phrase pair, they only concern the *overlap subhypergraphs* which may generate the alignments that are not consistent with the phrase pair. As an example, consider the phrase pair in the grey shadow in Figure 3(a), it is consistent with all sub-alignments from both h_1 and h_2 because they are outside and inside the phrase pair respectively, while not consistent with the sub-alignment that contains hyperedge e_2 from $h3$ because it contains an alignment link that crosses the phrase pair. Liu et al., [4] show that nearly 90% of the phrase pairs only need to concern less than 20% of all subhypergraphs, suggesting that the enumeration of all consistent alignments for each rule is practically feasible.

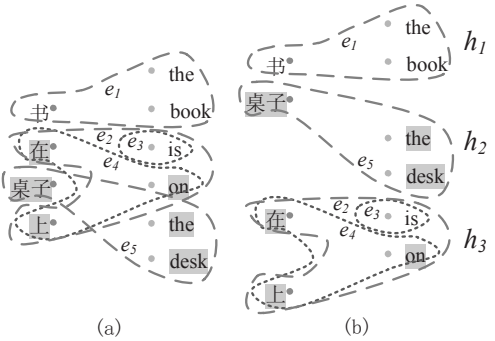

Fig. 3. (a) An example of a hypergraph in which the nodes in the grey shadow are the candidate phrase, (b) the independent subhypergraphs of the hypergraph in (a).

2.3 Drawbacks

For each overlap subhypergraph, Liu et al., [4] first enumerate all possible alignments using Algorithm 1, then check whether the translation rule is consistent with each alignment. This method has two drawbacks:

1. It is memory-intensive. As the number of alignments is exponential (although enumerable) in each hypergraph, it will consume a lot of memory to store all possible alignments, especially for the hypergraphs that have relatively many hyperedges.
2. It is time-consuming. To check consistence between an alignment and a rule that contains non-terminals (sub-phrase pairs in a phrase pair are replaced with non-terminals), they should check the consistency between the alignment and all phrase pairs (including the sub-phrase pairs). It will be time-consuming and not applicable to the rules that contain many non-terminals (e.g., SCFG rules in syntax-based translation models).

Given the great number of translation rules extracted from the hypergraphs, it will take most of the time to enumerate all consistent alignments for each rule. Therefore, we propose a simple and fast strategy to speed up the process, and make the weighted alignment hypergraph applicable to the syntax-based translation models.

3 Optimization

We borrow ideas from the inside-outside algorithm that successfully works in compact representations [5,9,10,7,8], and apply it to the weighted alignment hypergraph.

Given a phrase pair and a overlap subhypergraph, we divide the hyperedges in the subhypergraph into three categories: (1) *inside hyperedges* that only cover the vertices inside the phrase pair, (2) *outside hyperedges* that only cover the vertices that outside the phrase pair, and (3) *crossed hyperedges* that cover the vertices both inside and outside the phrase pair. Take the overlap subhypergraph h_2 in Figure 3(b) as example, e_4 is an inside hyperedge, e_3 is an outside hyperedge, and e_2 is a crossed hyperedge.

If we remove the crossed hyperedges from a overlap subhypergraph, we will obtain two independent partial hypergraphs: *inside partial hypergraph* whose vertices are connected by the inside hyperedges, and *outside partial hypergraph* whose vertices are connected by the outside hyperedges. Given a phrase pair P, the probability sum of all consistent sub-alignments encoded in the overlap subhypergraph h is:

$$\begin{aligned} p(A|h,P) &= \sum_{a \in A} p(a|h,P) \\ &= \sum_{a \in A_I} p(a|IPH) \times \sum_{a \in A_O} p(a|OPH) \end{aligned} \tag{2}$$

Here IPH and OPH denote the inside and outside partial hypergraphs respectively, and A_I and A_O denote the sub-alignments generated from them individually. Let OS denotes the set of overlap subhypergraphs for the phrase pair, then

$$p(A|H,P) = \prod_{h_i \in OS} p(A|h_i,P) \times \prod_{h_i \in H-OS} p(A|h_i) \quad (3)$$

Here the set of non-overlap subhypergraphs ($H-OS$) are irrelevant to the phrase pair, and we have $p(A|h,P) = p(A|h)$ for each $h \in H-OS$. Then the fractional count of the phrase pair is:

$$count(P|H) = \frac{p(A|H,P)}{p(A|H)} = \frac{\prod_{h_i \in OS} p(A|h_i,P)}{\prod_{h_i \in OS} p(A|h_i)} \quad (4)$$

We can easily extend this process to variable rules that contain non-terminals. For example, we have two sets of overlap subhypergraphs for a variable rule that contains one non-terminal (i.e., the phrase pair and the sub-phrase pair). Note that the subhypergraphs intersection set overlaps both the phrase and the sub-phrase. Therefore, we divide the hyperedges in the intersection set into four categories: (1) inside the sub-phrase pair, (2) outside the sub-phrase pair but inside the phrase pair, (3) outside the phrase pair, and (4) crossed hyperedges. Then we replace the two factors in Eq. 2 with the first three categories above. We use Eq. 2 for the other overlap subhypergraphs for the phrase and sub-phrase pairs respectively.

The advantage of inside-outside algorithm is we can employ shared structures for efficient dynamic programming. For example, when enumerating consistent alignments for a rule containing one non-terminal in Bull's algorithm, we should repeatedly check the consistency between the alignments and the phrases (sub-phrases). With inside-outside algorithm, we only need to concern the varied part of the structure (i.e. the intersection between the phrase and the sub-phrase), and re-used the previous calculated probabilities of the shared structures (i.e. the inside probability of the sub-phrase and the outside probability of the phrase). This greatly speeds up calculating probabilities of rules that contains non-terminals (§ 4.2).

4 Experiments

4.1 Setup

We carry out our experiments using a reimplementation of the hierarchical phrase-based system [2]. Each translation rule is limited to have at most two non-terminals. Our training data is FBIS corpus from LDC dataset that contains 239K sentence pairs.[1] We first follow Venugopal et al. [11] to produce n-best

[1] The FBIS corpus shares similar subhypergraph distribution with a larger corpus that contains 1.5 million sentence pairs. We believe that our results also suits large-scale corpora.

lists via GIZA++. We produce 20-best lists in two translation directions, and use "grow-diag-final-and" strategy [3] to generate the final 100-best lists by selecting the top 100 alignments. Finally we construct weighted alignment hypergraphs from these 100-best lists. For computational tractability, we follow Liu et al. [4] to only allow a subhypergraph has at most 10 hyperedges.

4.2 Results

Figure 4 shows the comparison results between the conventional Bull's algorithm (Bull) and our optimized approach (Optimization). We find that the optimization spends half of the time compared with Bull's algorithm, indicating that our approach speeds up the rule extraction.

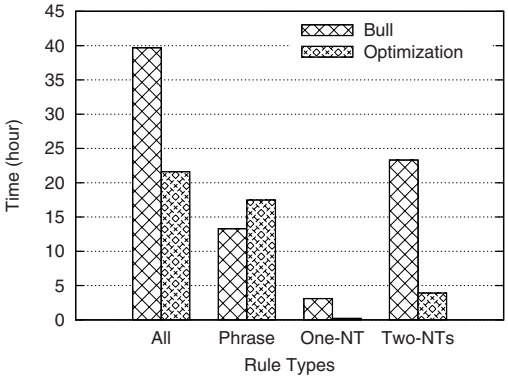

Fig. 4. The comparison results of two approaches

For both approaches, phrase extraction (rules without non-terminals) consumes a high portion of time. This is in accord with intuition, because we extract all possible candidate phrases from the hypergraphs. To maintain a reasonable rule table size, we only remain more promising candidates that have a fractional count higher than a threshold, which are used to generate rules with non-terminals. It should be emphasized that our approach consumes more time on phrase extraction, because we need to calculate the alignments probability sums for both inside and outside hyperedges, which can be reused in the calculation of rules with non-terminals.

Concerning translation rules with non-terminals (i.e. One-NT and Two-NT), we see that Bull's algorithm spends much more time than our approach. In Bull's algorithm, each alignment is repeatedly checked even they share many sub-alignments. To make things worse, for a rule with n non-terminals, each alignment is checked (n+1) times (the phrase and the n sub-phrases). In contrast, our approach only need to concern the intersection of the phrase and the non-terminals. This proves that our approach is applicable to SCFG rules that contain many non-terminals.

4.3 Analyses

Why Our Approach Is Faster?. From Figure 4 we can see that our approach outperforms Bull's algorithm mainly due to we spend much less time on the rules with two non-terminals. In this section, we will investigate that why our approach is faster on the rules with two non-terminals. For the intersecting overlap subhypergraphs of the two sub-phrases (the two phrase that are replaced with non-terminals), Bull's algorithm should check whether the alignments generated from these subhypergraphs are consistent with the the phrase and the two sub-phrases. Figure 5(Bull) shows the comparison of two approaches on the rules with two non-terminals. We find that nearly half of the subhypergraphs have no less than 5 alignments, which makes the Bull's algorithm very time-consuming.

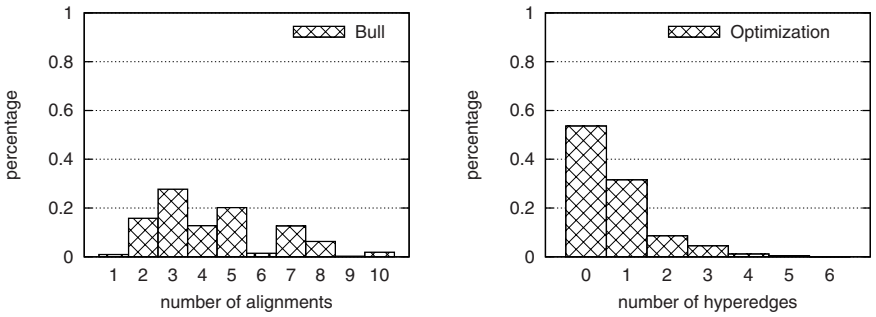

Fig. 5. The comparison of two approaches on the rules with two non-terminals

In contrast, our approach only focuses on the crossed hyperedges that may generate alignments that are not consistent with the rule. For example, given an overlap subhypergraph that overlaps both two sub-phrases, its vertices can be divided into three parts: (1) inside the first sub-phrase, (2) inside the second sub-phrase, (3) outside the two sub-phrases. Therefore, we divide the hyperedges in the subhypergraph into four categories: (1) inside the first sub-phrase, (2) inside the second sub-phrase, (3) outside the two sub-phrases, and (4) crossed hyperedges. The first two sets are previously calculated and we only need to concern (3): the hyperedges outside the two sub-phrases. Figure 5(Optimization) shows the distribution of the hyperedges outside the two sub-phrases. It should be noted that if there is no hyperedges outside the two sub-phrases, no sub-alignments can be constructed to cover the vertices outside the two sub-phrases. Then all the alignments generated from this subhypergraph are not consistent with the rule, and we can directly filter this rule without any calculation. Figure 5 shows that half of the overlap subhypergraphs have no hyperedges outside the two sub-phrases, which will greatly avoid the time-consuming calculation. Even in the rest subhypergraphs, most of them have no more than 3 hyperedges, which will spend few time for the calculation.

In conclusion, our approach avoids a great number of unnecessary and repeated calculation, and speeds up the process.

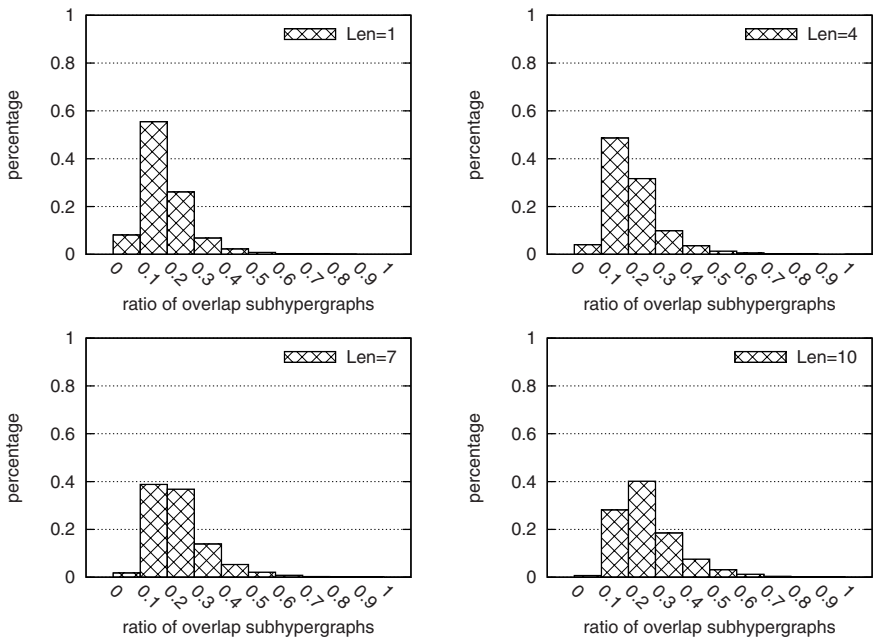

Fig. 6. The influence of phrase length on the ratio of overlap subhypergraphs to all subhypergraphs

The Influence of Phrase Length. As aforementioned, the percentage of overlap subhypergraphs in all subhypergraphs determines the efficiency of rule extraction. As one would expect, there are more overlap subhypergraphs for longer phrases, because a subhypergraph is more likely to cross the phrase and becomes an overlap subhypergraph for the phrase. Figure 6 shows the influence of phrase length on the proportion of overlap subhypergraphs, where xtics denotes the ratio of overlap subhypergraphs to all subhypergraphs and higher value means more overlap subhypergraphs need to be processed. We can see that the ratio of overlap subhypergraphs goes up with the increase of phrase length. For example, when the phrase length is 1 (the top left figure), nearly 90% of the phrase pairs only need to concern less than 30% of all subhypergraphs. When the phrase length increases to 10 (the bottom right figure), more than 30% of the phrase pairs need to process more than 30% of all subhypergraphs. The results indicate that the limit on the phrase length will speed up the rule extraction.

5 Conclusion and Future Work

In this paper, we propose a inside-outside algorithm to speed up the rule extraction in weighted alignment hypergraphs. Experimental results shows that the dynamic programming algorithm is twice as fast as the Bull's algorithm, and makes our approach applicable to syntax-based translation models.

Currently, we still spend much time on the calculating probabilities of phrases, which are reused in the calculation of rules with non-terminals. In the future, we will develop an efficient algorithm to speed up this process.

Acknowledgement. The authors are supported by 863 State Key Project No. 2011AA01A207, National Key Technology R&D Program No. 2012BAH39B03 and National Natural Science Foundation of China (Contracts 61202216). Qun Liu's work is partially supported by Science Foundation Ireland (Grant No.07/CE/I1142) as part of the CNGL at Dublin City University.

References

1. Brown, P.E., Pietra, S.A.D., Pietra, V.J.D., Mercer, R.L.: The mathematics of statistical machine translation: Parameter estimation. Computational Linguistics 19(2), 263–311 (1993)
2. Chiang, D.: Hierarchical phrase-based translation. Computational Linguistics 33(2), 201–228 (2007)
3. Koehn, P., Och, F.J., Marcu, D.: 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, vol. 1, pp. 48–54. Association for Computational Linguistics (2003)
4. Liu, Q., Tu, Z., Lin, S.: A Novel Graph-based Compact Representation of Word Alignment. In: Proceedings of the 51th Annual Meeting of the Association for Computational Linguistics (2013)
5. Liu, Y., Xia, T., Xiao, X., Liu, Q.: Weighted alignment matrices for statistical machine translation. In: Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, pp. 1017–1026. Association for Computational Linguistics, Singapore (2009)
6. Moore, R.C.: A discriminative framework for bilingual word alignment. In: Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, pp. 81–88. Association for Computational Linguistics, Vancouver (2005)
7. Tu, Z., Jiang, W., Liu, Q., Lin, S.: Dependency Forest for Sentiment Analysis. In: Zhou, M., Zhou, G., Zhao, D., Liu, Q., Zou, L. (eds.) NLPCC 2012. CCIS, vol. 333, pp. 69–77. Springer, Heidelberg (2012)
8. Tu, Z., Liu, Y., He, Y., van Genabith, J., Liu, Q., Lin, S.: Combining Multiple Alignments to Improve Machine Translation. In: Proceedings of the 24th International Conference on Computational Linguistics (2012)

9. Tu, Z., Liu, Y., Hwang, Y.-S., Liu, Q., Lin, S.: Dependency forest for statistical machine translation. In: Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010), pp. 1092–1100. International Committee on Computational Linguistics, Beijing (2010)
10. Tu, Z., Liu, Y., Liu, Q., Lin, S.: Extracting Hierarchical Rules from a Weighted Alignment Matrix. In: Proceedings of 5th International Joint Conference on Natural Language Processing, pp. 1294–1303. Asian Federation of Natural Language Processing, Chiang Mai (2011)
11. Venugopal, A., Zollmann, A., Smith, N.A., Vogel, S.: Wider pipelines: n-best alignments and parses in mt training. In: Proceedings of AMTA, Honolulu, Hawaii (2008)