An Algorithm for Word-Level Alignment of Parallel Dependency Trees¹

Yuan Ding Daniel Gildea Martha Palmer

Department of Computer and Information Science University of Pennsylvania Philadelphia, USA {yding, dgildea, mpalmer}@linc.cis.upenn.edu

Abstract

Structural divergence presents a challenge to the use of syntax in statistical machine translation. We address this problem with a new algorithm for alignment of loosely matched non-isomorphic dependency trees. The algorithm selectively relaxes the constraints of the two tree structures while keeping computational complexity polynomial in the length of the sentences. Experimentation with a large Chinese-English corpus shows an improvement in alignment results over the unstructured models of (Brown et al., 1993).

1 Introduction

The statistical approach to machine translation, pioneered by (Brown et al., 1990, 1993), estimates word to word translation probabilities and sentence reordering probabilities directly from a large corpus of parallel sentences. Despite their lack of any internal representation of syntax or semantics, the ability of such systems to leverage large amounts of training data has enabled them to perform competitively with more traditional interlingua based approaches. In recent years, hybrid approaches, which aim at applying statistical models to structural data, have begun to emerge. However, such approaches have been faced with the problem of pervasive structural divergence between languages, due to both systematic differences between languages (Dorr, 1994) and the vagaries of loose translations in real corpora.

Syntax-based statistical approaches to alignment began with (Wu, 1997), who introduced a polynomial-time solution for the alignment problem based on synchronous binary trees. (Alshawi et al., 2000) extended the tree-based approach by representing each production in parallel dependency trees as a finite-state transducer. Both these approaches learn the tree representations directly from parallel sentences, and do not make allowance for non-isomorphic structures. (Yamada and Knight, 2001, 2002) model translation as a sequence of operations transforming a syntactic tree in one language into the string of the second language, making use of the output of an automatic parser in one of the two parallel languages. This allows the model to make use of the syntactic information provided by treebanks and the automatic parsers derived from them. While we would like to use syntactic information in both languages, the problem of non-isomorphism grows when trees in both languages are required to match. The use of probabilistic tree substitution grammars for tree-to-tree alignment (Hajic et al., 2002) allows for limited non-isomorphism in that n-to-m matching of nodes in the two trees is permitted. However, even after extending this model by allowing cloning operations on subtrees, (Gildea, 2003) found that parallel trees overconstrained the alignment problem, and achieved better results with a tree-to-string model using one input tree than with a tree-to-tree model using two.

In this paper we present a new approach to the alignment of parallel dependency trees, allowing the tree structure to constrain the alignment at the high level, but relaxing the isomorphism constraints as necessary within smaller subtrees. The algorithm introduced in this paper addresses the alignment

¹ This work is supported by National Science Foundation under grant No. 0121285. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

task on its own, rather than viewing it as a part of a generative translation model. The algorithm uses the parallel tree structure to iteratively add constraints to possible alignments between the two input sentences. In each iteration, the algorithm first estimates the word-to-word translation parameters, then computes the scores of a heuristic function for each partial alignment, fixes the more confident partial alignments, and re-estimates the translation parameters in the next iteration.

We first give a formal definition of the alignment problem and introduce the iterative framework of the algorithm in section 2. Then we explain the usage of dependency trees and two heuristic functions in sections 3 and 4. The formal algorithm and an example for illustration are given in sections 5 and 6. The alignment results are evaluated in section 7.

2 The Framework

2.1 The Alignment Problem

Mathematically, the machine translation task can be viewed as a noisy channel. Given the task of translating a foreign language sentence to English,

we have: $P(e \mid f) = \frac{P(e)P(f \mid e)}{P(f)}$ where f and e are source and target language sentences. We want to find $e^* = \underset{e}{\operatorname{arg\,max}} P(e)P(f \mid e)$. The conditional probability part of the right hand side is usually referred to as the translation model (TM). During the construction of a machine translation pipeline, the alignment problem is usually handled as part of the TM and $P(f \mid e) = \sum_{a} P(f, a \mid e)$,

where a is any possible alignment between e and f. This approach requires a generative translation model.

However, when the alignment problem is viewed on its own, a generative model is not necessary. In other words, we can simply maximize P(a | e, f)using a conditional model. More straightforwardly, the alignment problem can be defined as Definition (1), which is equivalent to the alignment problem definition in (Och and Ney, 2000):

Here, f_j and e_{a_j} are words in the source and target language sentences f and e, respectively. 0

Definition (1) For each $f_j \in f$, find a labeling e_{a_j} , where $e_{a_j} \in e \cup \{0\}$

stands for the "empty symbol", which means f_j could be aligned to nothing. This definition does not allow multiple English words being aligned to a same foreign language word.

2.2 Algorithm Outline

We introduce the framework of the alignment algorithm by first looking at how the IBM models handle alignment. In Model 1, all connections for each foreign position are assumed to be equally likely, which implies that the orders of the words in both sentences do not matter. Model 2 more realistically assumes that the probability of a connection depends on the position which it connects to and on the length of the two strings. In Models 3, 4 and 5, the foreign language string is developed by choosing for each word in the English sting, first the number of words in the foreign language string it will generate, then the identity of each foreign language word, and finally the positions that these foreign language words will Briefly, Model 1 removes all the occupy. positioning information from the sentence pair and can be viewed as a "bag of words" model. And starting with Model 2, word positioning information is gradually added.

We do not have to model word positioning information the way Models 2 to 5 did if we can find an alternative source for positioning information. Syntactic structures offer a promising We use automatic syntactic parsers to choice. produce the parallel unaligned syntactic structures: (Collins, 1999) for English, (Bikel, 2002) for Chinese. The syntactic trees model the interaction between words within one language, instead of viewing them as a linear sequence. If two syntactic trees are partially aligned, the freedom of alignment for the rest of the unaligned nodes in the two trees is restricted by the aligned nodes, and these unaligned nodes are no longer free to be aligned to any position in the counterpart sentence.

Observing this, an iterative algorithm for MT alignment that bootstraps between statistical modeling and the parallel tree structures can be constructed. The following description of the

algorithm views the alignment problem as a labeling task as mentioned in Definition (1):

- **Step 0.** Unfix all the labeling for f_j except the root; initialize the two trees for *e* and *f* as the only tree pair.
- **Step 1.** Train a statistical translation model on all the tree pairs to acquire word-to-word translation probabilities. The tree pairs are used as "bags of words" in this step and a statistical model such as IBM Model 1 can be used.
- **Step 2.** Compute the labeling for all unfixed nodes. Use a heuristic function to select confident labelings, and merge these labelings into the fixed labeling set.
- **Step 3.** Project the fixed labeling set back onto the English and foreign language tree structures as fixed nodes
- **Step 4.** Partition the both the English and foreign language trees with the fixed nodes, producing a set of "treelet" pairs.
- Step 5. Go to Step 1 unless enough nodes fixed

We call the resulting trees from a tree partitioning operation "treelets" instead of "subtrees" since they do not necessarily go down to every leaf. An example of a tree partition operation is shown in Figures 1a and 1b.



Figure 1a "The quick brown fox jumps over the lazy dog."



Figure 1b Partitioning by fixing "*fox*" "*dog*" and "*jump*"

In each iteration of the algorithm, some unfixed node pairs are fixed. As the algorithm walks through the iterations, we have an increasing number of fixed node pairs, together with increasingly fine-grained treelet pairs in step 4. The algorithm stops when enough labels are fixed. Overall, each foreign word is free to be aligned to any English word initially. As the algorithm continues, both the foreign and the English treelets get smaller in size and the complexity of labeling is reduced.

3 Using Dependency Trees

This framework of alignment utilizes tree structures to provide the positioning information. At a glance, phrasal structure trees (treebank style trees) seem to be a natural choice. However, phrasal structure trees have two types of nodes, namely nonterminals and lexical items. The projected labelings always go to the lexical items while the domain information is stored in the nonterminals. It is difficult to determine how to partition a phrasal structure tree. To solve this problem, we use dependency trees.

In addition, the dependency representation has the best phrasal cohesion properties. The percentage for head crossings per chance is 12.62% and that of modifier crossings per chance is 9.22%, according to (Fox, 2002).

Dependency trees can be constructed from phrasal structure trees using a head percolation table (Xia, 2001). Each node of a dependency tree dominates its descendents. If the dependency tree is stored with local word order information at its nodes, a tree traversal algorithm starting at any arbitrary node of the tree always generates a chunk in the original sentence. The sentence "*The quick brown fox jumps over the lazy dog*" is represented as a dependency tree in Figure 1a.

When some of the nodes in a dependency tree are fixed, we can partition the tree with each of the fixed nodes as a new root for the resulting treelets. Figure 1b shows the resulting three treelets by fixing the words "*jumps*", "*fox*" and "*dog*". Algorithmically, fixing one node in a treelet will result in two subsequent treelets. The treelet rooted at the original root is called the "outside treelet" and the treelet rooted at the newly fixed node is called the "inside treelet". This is shown in Figure 2. As a result, if we have n fixed nodes in a dependency tree, we will always have n treelets (assuming the

root of the whole tree is always fixed). This guarantees that whenever we fix a pair of nodes, the resulting partitioned treelets are always matched in pairs.



Figure 2 Inside and Outside Treelets Partition "*a recently built house*" by fixing "*built*"

The correctness of the algorithm lies in what we refer to as our *partition assumption*:

Partition Assumption If (f_j, e_{a_j}) $(f_{j'}, e_{a'_j})$ are two aligned pairs of nodes in two languages and $f_{j'}$ is a descendent of f_j in the dependency tree of f, $e_{a'_j}$ must also be a descendent of e_{a_j} in the dependency tree of e

This assumption is much looser than requiring the two dependency trees of the two languages to be isomorphic. A typical violation of this assumption is the set of crossing-dependency alignments between two structures shown in Figure 3 below:



Figure 3. A crossing-dependency alignment (dotted lines stand for the alignment)

While such violations of the partition assumption exist in real language pairs (as reported by (Fox, 2002) to be around 10%), the hope is, if we only fix node pairs above and below such violations, the "bag of words" translation model will correctly align the words with crossing-dependency alignments.

4 Heuristics

The heuristic function in Step 2 takes a tentative node pair (f_i, e_{a_i}) from the two treelets and

outputs a certain value which corresponds to confidence in this labeling. Here we introduce two heuristic functions.

4.1 Entropy

Since the label for the foreign language word f_j is computed by simply choosing $\underset{e_i \in E}{\operatorname{arg max}} t(f_j | e_i)$, where E is the set of possible labels for f_j in the English treelet, and $t(f_j | e_i)$ is the probability of e_i being translated to f_j , we have $t(e_i | f_j) = \frac{t(f_j | e_i)P(e_i)}{P(f_j)}$. A reliable set of

translation probabilities will provide a distribution with a high concentration of probabilities on the chosen labels. Intuitively, the conditional entropy of the translation probability distribution will serve as a good estimate of the confidence in the chosen labeling. Let $S = \sum_{e_i \in E} t(e_i \mid f_j)$ and define $\hat{e} \in E$ as

a random variable.

$$\begin{split} H(\hat{e} \mid f_{j}) \\ &= \sum_{e_{i} \in E} -\frac{t(e_{i} \mid f_{j})}{S} \log(\frac{t(e_{i} \mid f_{j})}{S}) \\ &= \frac{\sum_{e_{i} \in E} -t(e_{i} \mid f_{j}) \log(t(e_{i} \mid f_{j}))}{S} + \log S \end{split}$$

Since we need to compute conditional entropy given f_j , here the translation probabilities are normalized. The first heuristic function is defined as:

$$h_1(f_i, e_i) = H(\hat{e} \mid f_i)$$

4.2 Inside-Outside Probability

Suppose we have two trees initially, T(e) and T(f). When we fix a pair of words (f_j, e_{a_j}) , both trees will be partitioned into two treelets. Here we borrow the idea from PCFG parsing, and call the treelets rooted with e_{a_j} and f_j inside treelets $T_{in}(e, e_{a_j})$ and $T_{in}(f, f_j)$. And the other two treelets are called outside treelets $T_{out}(e, e_{a_j})$ and $T_{out}(f, f_j)$. So, we have:

$$P(T(f), (f_{j}, e_{a_{j}})|T(e))$$

= $P(T_{out}(f, f_{j}) | T_{out}(e, e_{a_{j}}))$
 $\times P(T_{in}(f, f_{j}) | T_{in}(e, e_{a_{j}}))$

Define the size function to be the number of nodes in a treelet. And let

$$l_{out} = size(T_{out}(e, e_{a_j})), m_{out} = size(T_{out}(f, f_j))$$
$$l_{in} = size(T_{in}(e, e_{a_j})), m_{in} = size(T_{in}(f, f_j))$$

we have:

$$P(T_{out}(f, f_{j}) | T_{out}(e, e_{a_{j}})))$$

$$= \frac{1}{(l_{out} + 1)^{m_{out}}} \prod_{f_{k} \in T_{out}(f, f_{j})} \sum_{e_{a_{k}} \in T_{out}(e, e_{a_{j}}) \cup \{0\}} t(f_{k} | e_{a_{k}}))$$

$$P(T_{in}(f, f_{j}) | T_{in}(e, e_{a_{j}}))$$

$$= \frac{1}{(l_{in} + 1)^{m_{in}}} \prod_{f_{k} \in T_{in}(f, f_{j})} \sum_{e_{a_{k}} \in T_{in}(e, e_{a_{j}}) \cup \{0\}} t(f_{k} | e_{a_{k}})$$

The above two probabilities are derived using ideas from IBM Model 1. Similarly, define the second heuristic function as:

 $h_2(f_j, e_{a_j}) = P(T(f), (f_j, e_{a_j})|T(e))$

4.3 Fertility Threshold

Again, let us suppose we want to create two treelet pairs by fixing the node pair (f_j, e_{a_j}) . In the first heuristic, the size and topology of the two resulting treelet pairs are not taken into consideration. If we are unlucky enough, we may well end up having a treelet pair with one node on one side and 9 nodes on the other. So we set a fertility threshold to make sure that the resulting treelet pairs have reasonable sizes on both sides. Currently it is set to be 2.0. Any labeling resulting in a treelet pair that the number of the nodes on one side is larger than twice the number of the nodes on the other will be discarded.

5 Formal Algorithm

Here we present the formal algorithm. The following invariant holds at Step 1 of each iteration because every new treelet created by the tree partition operation in Step 3 is an inside treelet rooted by a fixed node. Hence it is guaranteed that each partition operation in Step 3 will create a new treelet pair.

An Invariant of the Algorithm:

At Step 1 of each iteration, $(e_{a_j}, f_j) \in Fixed(e, f)$ if and only if exists $(T_e, T_f) \in Trees(e, f)$ such that $f_j = root(T_f)$ and $e_{a_j} = root(T_e)$

• Step 0:

Let $Fixed(e, f) = \{(root(T(e)), root(T(f)))\};$ Let $Trees(e, f) = \{(T(e), T(f))\}$ where T(e) and T(f) are dependency structures constructed from e and f.

• Step 1:

Run IBM Model 1 on Trees(e, f) to acquire translation probability $t(f_j | e_i)$ for alignment

• Step 2:

L

For each $(T_e, T_f) \in Trees(e, f)$ {

Let $Unfixed(T_e)$

$$= \{e_i \mid e_i \in T_e, (e_i, *) \notin Fixed(e, f)\}$$

et
$$Unfixed(T_f)$$

$$= \{ f_j \mid f_j \in T_f, (*, f_j) \notin Fixed(e, f) \}$$

For each $f_i \in Unfixed(T_f)$ {

$$e_{a_{j}} = \underset{e_{i} \in Unfixed(T_{e})}{\arg \max t(f_{j} | e_{i})}$$

if $(h(f_{j}, e_{a_{j}})$ and $(f_{j}, e_{a_{j}})$ satisfy certain
constraints), Let
Fixed $(e, f) = Fixed(e, f) \cup (e_{a_{i}}, f_{j})$

}// for each}// for eachStep 3:

For each $(T_e, T_f) \in Trees(e, f)$ { if (*Exists* $(f_i \in T_f, e_{a_i} \in T_e, (f_i, e_{a_i}) \in Fixed(e, f)))$

$$Trees(e, f) = Trees(e, f) \cup \left\{ \left(T_{inside}(T_e, e_{a_j}), T_{inside}(T_f, f_j) \right) \right\} \cup \\ \left\{ \left(T_{outside}(T_e, e_{a_j}), T_{outside}(T_f, f_j) \right) \right\}$$

} //if } // for each

• Step 4: goto Step 1 unless enough nodes fixed

In IBM Model 1, P(f | e) has a unique local maximum, so the values in the *t* table are independent of initialization. In Step 1, the values in *t* table converge to a unique set of values. So theoretically, the result of this algorithm is only dependent on the choice of the heuristic function *h*. The algorithm calls the heuristic function *h* linearly to the size of the treelet. So the time complexity of the algorithm is $O(n \times T(h))$, where T(h) is the time complexity for the heuristic function *n* is the length of the shorter sentence of the sentence pair.

6 An Example

An example of iterative partitioning of the treelets is given below to illustrate the algorithm. The Chinese is given in romanised form.

- [English] I have been here since 1947.
- [Chinese] 1947 nian yilai wo yizhi zhu zai zheli.









Figure 4. An Example

7 Evaluation

We used the LDC Xinhua newswire Chinese – English parallel corpus with 60K+ sentence pairs as the training data.¹ The parser generated 53130 parsed sentence pairs. We evaluated different alignment models on 500 sentence pairs provided by Microsoft Research Asia. The sentence pairs are word level aligned by hand.

In Tables 1 and 2, the numbers listed are the F-scores for the alignments. $F = \frac{2|A \cap G|}{|A|+|G|}$,

where A is the set of word pairs aligned by the automatic alignment algorithm and G is the set of word pairs aligned in the gold file.

In Table 1, the IBM models are bootstrapped from Model 1 to Model 4. The evaluation F-scores did not show significant improvement from Model 1 to Model 4, which we believe is partially caused by the difference in the genres of the training and evaluation data. Also the IBM models showed signs of overfitting.

Itn#	IBM 1	IBM 2	IBM 3	IBM 4
1	0.0000	0.5128	0.5082	0.5130
2	0.2464	0.5288	0.5077	0.5245
3	0.4607	0.5274	0.5106	0.5240
4	0.4935	0.5275	0.5130	0.5247
5	0.5039	0.5245	0.5138	0.5236
6	0.5073	0.5215	0.5149	0.5220
7	0.5092	0.5191	0.5142	0.5218
8	0.5099	0.5160	0.5138	0.5212
9	0.5111	0.5138	0.5138	0.5195
10	0.5121	0.5127	0.5132	0.5195
N 11 4	F 1 (1.		x 1 1 1

 Table 1: Evaluation results for IBM Models 1-4

¹ The original LDC Xinhua newswire corpus is very noisy and we filtered out roughly half the sentences.

In Table 2, Models h1 and h2 are our models that use heuristic function h1 and h2, respectively. We find that with the current parameter settings, Models h1 and h2 tend to overfit after the second iteration. Table 2 shows results after one iteration of steps 2 through 4 of our algorithm, after successive iterations of IBM Model 1 in step 1 of the second iteration. The two iterations of the algorithm take less than two hours on a dual Pentium 1.2GHz machine for both heuristics. Error analysis showed that the overfitting problem is mainly caused by violation of the partition assumption in fine-grained dependency structures.

M1 Itn#	Model h1	Model h2
1	0.5549	0.5151
2	0.5590	0.5497
3	0.5632	0.5515
4	0.5615	0.5521
5	0.5615	0.5540
6	0.5603	0.5543
7	0.5612	0.5539
8	0.5604	0.5540
9	0.5611	0.5542
10	0.5622	0.5535

Table 2: Evaluation results for our algorithm with
heuristic function h1 and h2

8 Conclusion

Our model, based on partitioning sentences according to their dependency structure, outperforms the unstructured IBM models on a large data set. The model can be thought of as in sense orthogonal to the IBM models in that it uses syntactic structure but no linear ordering information.

Possible future extensions to the model include adding information on linear order, allowing alignments that violate the partition assumption at some cost in probability, and conditioning alignment probabilities on part of speech tags.

9 References

- Hiyan Alshawi, Srinivas Bangalore, and Shona Douglas. 2000. Learning dependency translation models as collections of finite state head transducers. *Computational Linguistics*, 26(1): 45-60.
- Daniel M. Bikel. 2002. Design of a multi-lingual, parallel-processing statistical parsing engine. In *Proceedings of HLT 2002*.

- Peter F. Brown, John Cocke, Stephen A. Della Pietra, Vincent J. Della Pietra, Frederick Jelinek, John D. Laferty, Robert L. Mercer, and Paul S. Roossin. 1990. A statistical approach to machine translation. *Computational Linguistics*, 16(2): 79-85, June.
- Peter F. Brown, Stephen A. Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. 1993. The mathematics of statistical machine translation: parameter estimation. *Computational Linguistics*, 19(2): 263-311.
- Michael John Collins. 1999. Head-driven Statistical Models for Natural Language Parsing. Ph.D. thesis, University of Pennsylvania, Philadelphia.
- Bonnie J. Dorr. 1994. Machine translation divergences: A formal description and proposed solution. *Computational Linguistics*, 20(4): 597-633.
- Heidi J. Fox. 2002. Phrasal cohesion and statistical machine translation. In *Proceedings of EMNLP-02*, pages 304-311
- Daniel Gildea. 2003. Loosely tree based alignment for machine translation. In *Proceedings of the 41th Annual Conference of the Association for Computational Linguistics* (ACL-03)
- Jan Hajic, et al. 2002. Natural language generation in the context of machine translation. Summer workshop final report, Center for Language and Speech Processing, Johns Hopkins University, Baltimore.
- Franz Josef Och and Hermann Ney. 2000. Improved statistical alignment models. *In Proceedings of the 38th Annual Conference of the Association for Computational Linguistics* (ACL-00), pages 440-447, Hong Kong, October.
- Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational Linguistics*, 23(3):3-403.
- Fei Xia. 2001. Automatic grammar generation from two different perspectives. Ph.D. thesis, University of Pennsylvania, Philadelphia.
- Kenji Yamada and Kevin Knight. 2001. A syntax based statistical translation model. In *Proceedings of the 39th Annual Conference of the Association for Computational Linguistics* (ACL-01), Toulouse, France.
- Kenji Yamada and Kevin Knight. 2002. A decoder for syntax-based statistical MT. In *Proceedings of the 40th Annual Conference of the Association for Computational Linguistics* (ACL-02), Philadelphia.