Computational Linguistics in the Netherlands Journal 2 (2012) 71–96 Submitted 08/2012; Published 12/2012

Transformation-based tree-to-tree alignment

Gideon Kotzé
University of Groningen, Groningen, the Netherlands

Abstract

Previous experiments suggest that a rule-based approach to tree alignment error correction serves to be an effective complement to statistical alignment. We show how, using relatively few features, an implementation of Brill's Transformation-Based Learning algorithm improves the results of a high precision model of the statistical aligner Lingua-Align. Using our system to correct already tree aligned data, we achieve balanced F-scores of 80.6 on our test set and 85.2 on our development test set. Using it as a tree aligner on word aligned data, our best F-scores using the same model amount to 78.7 and 83.0 respectively. Finally, we apply a pipeline of alignment and error correction tools to create several versions of a large parallel treebank consisting of various domains for Dutch to English for use in a syntax-based MT system. We conclude that transformation-based learning is a promising approach for the large-scale creation of parallel treebanks for various NLP purposes.

1. Introduction

Various systems have been developed for non-terminal constituent alignment, either making use of hand-crafted rules (Menezes and Richardson 2001, Groves et al. 2004) or fully unsupervised methods (Zhechev and Way 2008, Lavie et al. 2008). The system Lingua-Align (Tiedemann 2010) is a supervised tree-to-tree aligner which implements discriminative models. In previous work, experiments suggest that a rule-based approach to tree alignment error correction can serve to be an effective complement to statistical alignment (Kotzé 2011a, Kotzé 2011b, Kotzé et al. 2012), using results from Lingua-Align as our testing ground. These affirm the good results obtained in the abovementioned rule-based systems. However, these experiments only make use of manual rule-based components with a limited scope. It would make sense to also be able to learn many rules that can be applied with much greater precision without the effort of constructing them manually.

Transformation-based learning (Brill 1995), TBL in short, has proven to be a simple yet powerful learning algorithm, applicable to a variety of tasks, also outside the field of NLP. Its error-driven nature makes it an ideal candidate for improving the performance of data sets which have already been processed, but it can also be successfully implemented as a classifying algorithm in itself. Furthermore, unlike with pure statistical algorithms, its rule-based nature provides for a very transparent error analysis procedure. We believe that in view of this, TBL is an excellent candidate for the problem of tree alignment error correction. To our knowledge, TBL has not yet been applied to this particular problem at the time of writing.

In this work, we design and implement our experiments in view of eventual application in syntax-based MT. However, we believe that the results of our experiments also provide some interesting linguistic insights. Error analysis can shed light on patterns regarding structural and lexical differences between the languages involved. Conversely, the most successful rules tell us which linguistic or structural generalisations can be made in the alignment process with a high rate of success.

In the next section, we introduce Brill's algorithm. After that, we present our adaptation to the problem in sections 3, 4, 5 and 6. This is followed, in section 7, by a presentation of the data and a description of the experiments. After a discussion of the results in section 8 we conclude this work in section 9.

2. Transformation-based learning

Transformation-based learning (TBL) is a supervised rule-based machine learning algorithm used to classify previously unseen examples of a data set. Ever since the introduction of the original concept by Eric Brill and its application on part-of-speech tagging (Brill 1992, Brill 1994, Brill 1995), there has been a flurry of publications on the topic, mainly on refining the algorithm (Samuel 1998, Ngai and Florian 2001), increasing the scope (Florian and Ngai 2001), or adapting it to different problems such as parsing (Brill 1993a, Brill 1993b, Brill 1996, Fung et al. 2004), prepositional phrase attachment disambiguation (Brill and Resnik 1994), base NP chunking (Ramshaw and Marcus 1995), spelling correction (Mangu and Brill 1997), dialogue act tagging (Samuel et al. 1998), phrase chunking (Florian et al. 2000), handwritten character segmentation (Kavallieratou et al. 2000), named entity recognition (Black and Vasilakopoulos 2002), semantic role labeling (Higgins 2004, Williams et al. 2004), word alignment (Ayan et al. 2005), newspaper headline generation (De Kok 2008) and syntactic tree transformation to improve convergence (Burkett and Klein 2012).

Although since the 90s there has been significant progress in optimizing and refining the algorithm, its basic premise remains unaltered: Training occurs by applying a series of rules to a data set. Each is applied separately in order to find the single best rule which, when applied, improves performance most. This is made possible by comparing the output to a gold standard. That rule is then marked as the best at that point and set aside. The data set is retained, with the changes made by the rule and other possible rules in earlier iterations, up to that point. Then all rules are again tested on the new and updated data set to find the next single best rule which improves performance the most. This continues until no rule can be found that improves the accuracy of the newest data set. The set of learned rules is essentially the model that is to be used on new unseen data.

Rules are generated by a pre-defined set of templates. A rule template can be defined as an abstraction of a rule which can be realised to more than one different instance. A triggering environment realises the rule when conditions specified in the environment hold. Generally, we can write it as an *if...then* statement:

if condition X then do Y

More specifically we can write this as the following example in the context of a tree alignment application:

if current node pair has tag combination X then change their link status to Y

Rule templates refer to features that can potentially describe useful relationships between elements in the data. In the example above, the relationship between the current and the previous tag is assumed to have the potential to be a useful feature in helping to determine the identity of the current tag.

After constructing the template list, we generate instantiations, which essentially form the list of all possible rules that can be applied. For the above tag template, this includes a list of possible tags to be included either as the previous or the current tag. Not all tag pairs may be useful, therefore the challenge lies in trying to find an optimal solution lying somewhere between brute force generation and more manual work (for example, using random sampling).

Often, learned rules are initially very general but become more specific as they approach the ground truth represented by the gold standard. An effective analogy is provided by Samuel (1998), who attributes it to Terry Harvey. A painter can decide, upon painting a barnyard scene, to first colour everything blue, since the sky will comprise the majority of the canvas area. After the paint dries, he paints the barnyard which comprises a smaller area, but without taking care to avoid the windows, roof and doors, which will be painted more precisely at a later stage. Similarly, the first few rules can be applied over a broad area, increasing recall but also making many mistakes. As more rules are applied, some mistakes are corrected and precision generally goes up.

With many TBL applications, the data is first annotated with a so-called *initial state annotator*. To explain this further it in terms of a part-of-speech tagging application, this can range from naively tagging all words as nouns to the output of an already accurate tagger with the intention of further increasing its accuracy. The object of the training phase is to learn the correct order of rules to get as close as possible to the *ground truth* represented by the gold standard.

3. Transformation-based learning for tree alignment

In this section, we describe our adaptation of TBL to the tree alignment problem. More specifically, the problem we are attempting to solve is the accurate alignment of non-terminal constituents between a pair of syntactically annotated parallel sentences. Minimally, we assume structurally valid phrase structure trees with existing word alignments. However, any additional metadata can be incorporated and used as features, such as dependency information, POS tags, category labels, lemmas, and so on. Instead of an alignment set that is only word aligned, one containing existing non-terminal alignments can also be used as an initial state annotator.

As discussed in the previous section, in TBL we apply a list of transformations consisting of a so-called *triggering environment* and a *rewrite rule*, which is the action undertaken when the required conditions specified by the triggering environment are met. In the case of tree alignment, there can always be only two general courses of action: add or remove an alignment. Whether an alignment can be added or removed depends on whether the currently inspected node pair is in fact aligned.

For every iteration, best rules are globally applied - currently, no feature extraction exists until the full data set has been updated by the rule. Adding or removing alignments may of course change the actual feature values of ancestors or descendents of particular node pairs, possibly leading to errors. However, assuming good feature selection and proper implemention, we can rely on later rules to correct these errors.

At the time of writing, we do not distinguish between adding different types of links, instead assuming that all links to be added are confident (good, not fuzzy - see section 4). Therefore, our rule templates consist solely of generalisations of possible conditions under which to add or remove an alignment. Since most existing links in our gold standards are confident, not adding fuzzy links does not significantly affect our results.

As mentioned before, a transformation can be represented in an abstract form called a *template*. The following is another example of a possible template in the context of tree alignment:

- if current node pair has a height difference of less than x, then change link type to y

where the link type can in this case only be ADD or REMOVE. However, the triggering environment may consist of a number of possible features, which we will discuss in the next section.

4. Features

Thanks to our experience in building training data and previous tree alignment experiments, we have identified a number of features that could be useful to act as triggering environments. We explain how we incorporate them into our system in section 5. All features are optional, although at least one is needed for training.

4.1 Tree features

4.1.1 Unary nodes

Some phrase-structure trees contain constituents that have only a single child node. Depending on the alignment scheme, one may prefer aligning either the parent or the child (or even the unary grandchild, if it exists). For example, in our gold standards we preferred to align the lowest possible node, since the information is most specific. For example, a VP can also be an S. Aligning the VP is more informative since there are also other types of S having different unary daughters.

Apart from checking if a specific node is in a unary relation, one may specify certain special conditions or constraints. For example, an S may be in a unary relationship with a ROOT node, apart from the fact that the ROOT node is also the parent of a full stop. One may decide that punctuation should be ignored in some contexts and make an exception in this case.

4.1.2 Subtree similarity

In Kotzé et al. (2012) we describe two key similarity features that we call leaf ratio similarity scores and linked leaf ratio similarity scores. Given a subtree pair, a leaf ratio similarity score is a measure of the ratio between the number of terminals (leaves) of the subtree with the least number of leaves and that of the subtree with the most number of leaves, as well as the leaf count difference between the two sides. The idea is that together, they constitute a measure of similarity, and the more similar the subtrees are, the more likely it is that their roots should be linked (given certain other conditions such as the condition that they share at least one word alignment). We have since also implemented using the geometric mean of the two numbers (ratio and count difference), which reflects the degree to which they share the same weight. It has the effect that the lesser score is weighted more heavily. If we use a penalty score, we assume that the weights are different. We attempt to model these differences using the value of the denominator.

A similar measure is calculated by the linked leaf ratio similarity score. In this case, the lowest value is the number of leaves in both subtrees that are linked, and the highest value is simply the total number of leaves in both subtrees. The idea behind this is that the more leaves are linked between two subtrees, the more likely it is that the trees should be linked. The ratios between these two numbers and the difference between them are used in exactly the same way as above to calculate two alternative ways of representing subtree similarity. The formulae for these measures are presented the aforementioned paper.

4.1.3 Height

We have found it useful to define the *height* of a node as the longest path from any of its leaves to the current node. Relative height difference between a given subtree pair could conceivably be a feature in helping to determine whether or not they should be linked. For example, it seems unlikely that a subtree at the first level above the terminal and spanning fewer words should be linked to a more complex subtree containing many recursive subtrees and typically spanning more words.

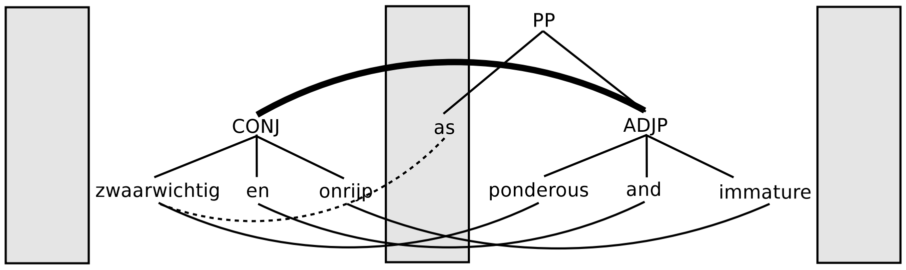
4.2 Link features

4.2.1 Link well-formedness

We use the *well-formedness* feature as a measure of whether or not there are links from any of the nodes in the subtree pair that are not shared by both of the trees. If one or more such "outgoing" links, which does not have the root of the other tree as an ancestor, exist, the trees are not well-formed with respect to its links. Our definition of well-formedness is based on the following axiom as stated by Hearne et al. (2007):

- A node in a tree may only be linked once.

Figure 1: An example of a non- (but nearly) well-formed linked subtree pair, indicated by the thick line connecting the nodes CONJ and ADJP. The fuzzy (and incorrect) alignment between "zwaarwichtig" and "as" is a less confident link leading to non-wellformedness of the subtrees, since "as" does not have ADJP as an ancestor. However, if we ignore this alignment and allow near-wellformedness, we can still link the trees in question. The grey areas represent "no go areas" for word alignments which render the subtrees non-wellformed whenever a descendent of any of the subtree roots is linked to a node there.



- Descendants of a source linked node may only be linked to descendants of its linked node on the target side and vice versa (target to source).

- Similarly, ancestors of a source linked node may only be linked to ancestors of its linked node on the target side and vice versa (target to source).[1]

In this paper, we will only refer to the last two points when mentioning well-formedness, as it is a significant indication of equivalence and an important discriminating feature.

As with the work presented in Zhechev and Way (2008), we distinguish between so-called *good* and *fuzzy* links, which denote confident and less confident links respectively. This distinction is used for a number of features described below.

Also in Kotzé et al. (2012), we have described the value of allowing for *near-wellformedness*. Restricting linking subtrees to only those that are strictly well-formed may lower recall significantly. Data sets containing high recall word alignments often have incorrect links between the terminals which should ideally not play a role in determining whether or not the subtree pair in question should be aligned. We have noticed that in some cases, allowing for one or two less confident word alignments to violate the principle increases both the recall and the resulting F-score. Figure 1 explains this graphically.

For our TBL system, it is possible to check for near-wellformedness in a number of possible ways:

- checking for n number of good outgoing links

- checking for n number of fuzzy outgoing links

- checking for n number of non-terminal outgoing links

- checking for all or any of the above, but ignoring punctuation - in other words, any links involving terminal nodes with punctuation are not counted

where by "outgoing" links we mean links between nodes where one of the nodes that are being linked has a relevant source or target subtree root node as an ancestor and the other one does not. For example, in figure 1, the subtrees CONJ and ADJP are linked. With this in mind, we note

1. In the literature, one side is often called the source side and the other the target side, even if eventual translation from the former to the latter is not the end application one has in mind.

that there is an outgoing link aligning *zwaarwichtig* with *as*. The word *zwaarwichtig* has a relevant ancestor - the parent node CONJ - whereas the word *as* has not, since ADJP, which is aligned to CONJ, is not an ancestor of *as*.

This can also easily be changed to count the number of nodes that have these links instead of the links themselves. In the case of multiply aligned nodes, these two numbers can differ significantly.

Based on manual inspection, we regard even one outgoing link from a non-terminal node to be significant evidence to discard a candidate node pair for alignment. Therefore, in our experiments, we did not allow for near-wellformedness in this case. However, it is still possible to implement this if one so chooses.

4.2.2 Link types

If a subtree pair shares only a few low probability (fuzzy) word alignments, it seems reasonable to conclude that perhaps the trees are not that equivalent to each other. The converse - that they are fairly equivalent if they share at least a few strong word alignments - seems to be true. Of course, the lack of any shared word alignments whatsoever would seem to be present strong evidence against the alignment of the subtree pair in question. We therefore include checking for the existence of a specified link type as a possible feature.

4.3 Orthographic features

It seems reasonable that if two strings are exactly equivalent, they might also be semantically equivalent, and should therefore be aligned, especially in the case of closely related languages. Additionally, if the span of a string includes the first or last token of the sentence, the same may be true of its translational equivalent.

One may additionally define a wide range of possible conditions or constraints such as the presence of only lower case letters, no punctuation, and so on.

4.4 Other features

4.4.1 Element on only one side

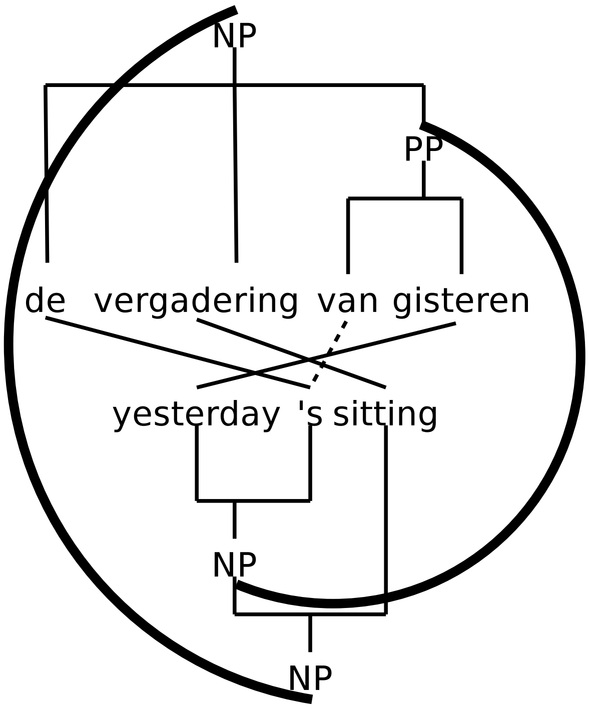
In a significant number of cases, we have found that if a verb occurs in either the source or target subtree but not on the other, they should not be aligned. The verb is conceivably the most important word class with which one describes an action. If one assumes the sentence pair to be a more or less equivalent translations of each other, it stands to reason that an action described on the one side must also occur on the other side. Of course, there are exceptions, and in many cases, the equivalent of a verb can be realized as another class such as a noun. However, it has still proven to be a useful discriminating feature. Similarly, other discriminating features such as numbers and capitalized words may also be used in a similar fashion.

4.4.2 Element at beginning or end of span

We have found that if any of the subtrees have a punctuation mark at either end, they are likely to be either at the beginning or end of the sentence or a clause, which in itself can be a useful discriminating feature.

Of course, this can be generalized to any element. Another useful element has proven to be the existence of an aligment that ensures well-formedness - in other words, an alignment from the beginning or end token that links to one on the other side that has the the root of the subtree as an ancestor. If this occurs on both source and target-sides and both at the beginning and end tokens, this seems to be a good indication, along with well-formedness, that the source and target-side spans are equivalent. If one side is not linked, this could mean that perhaps an ancestor ought to be linked instead. Figure 2 provides an example.

Figure 2: An example of an equivalent subtree pair where the first and last words of both sides (*de*, *gisteren*, *yesterday* and *sitting*) are linked to a descendent word of the respective other side. The Dutch phrase literally means *the sitting of yesterday*. The dotted word alignment indicates a fuzzy alignment, while the solid lines are confident.



5. Implementation

Given a subtree pair, a specific feature is either applicable or not, meaning that we have a hypothesis space consisting of boolean values with the size of 2^n where n is the number of features. Furthermore, features themselves may contain a possibly infinite range of values. For example, one may check for a height difference of 4, or 5, or 6, ad infinitum. This is only restrained by its practical application, which differs depending on the nature of the data being processed.

Because of the complexity of the task at hand, we have not yet implemented the generation of instantiations from feature templates. Instead, we manually experimented using various combinations of features and values. Training input currently consists of the two alignment sets (automatic and gold standard) as well as a manually produced instantiation written in a specific syntax.

Training initiates by extracting various features from both treebanks and the automatic STA-XML alignment file. The features are specified by the user in a text file using a special syntax. These are used to assign rule statistics to all possible non-terminal node pairs. The rule statistics essentially assign a profile of boolean feature values to every node pair according to the specified rule instantiation. In other words, every unique set of binary numbers refers to the feature profile of the node pairs having those values. These have the potential to function as triggering environments during actual rule application.

Figure 3 illustrates an example of the deletion of a link using a remove rule. The tree pair in question is the Dutch NP (s5_4) which is erroneously linked to the English VP (s5_503). The NP dominates *de schep* (*the shovel*) while the VP dominates *Bring the shovel.* These two phrases are clearly not equivalent. A remove rule associated with a specific set of feature values could lead to such cases being identified and removed. In a later stage, an addition rule with the feature values of the correct NP pair (indicated by the dashed line) could lead to these tree pairs being correctly linked.

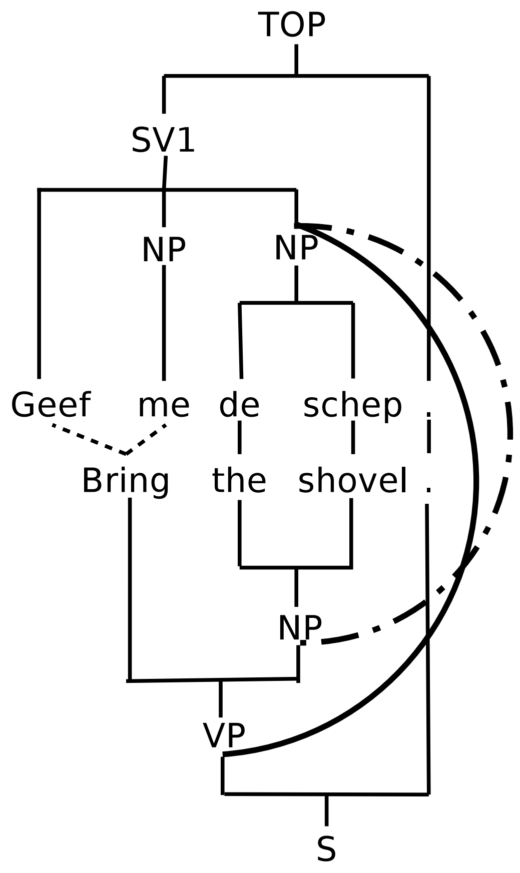
The following feature values could be extracted from the abovementioned node pair to be associated with a remove rule:

Node pair (s5_4)(s5_503) has the following properties:

- source node has a unary child = FALSE
- target node has a unary child = FALSE
- source node has a unary child, ignoring children with punctuation = FALSE
- target node has a unary child, ignoring children with punctuation = FALSE
- less than 1 confident (good) word alignment(s) going outside = TRUE
- less than 3 less confident (fuzzy) word alignment(s) going outside = TRUE
- less than 1 non-terminal alignment(s) going outside = FALSE
- less than 1 confident (good) word alignment(s) going outside, ignoring punctuation = TRUE
- less than 3 less confident (fuzzy) word alignment(s) going outside, ignoring punctuation = TRUE
- leaf ratio similarity score is less than 40; penalty score denominator is 80 = FALSE
- height difference is less than 4 = TRUE
- verb exists on only one side = TRUE
- pronoun exists on only one side = FALSE
- subtrees share at least one confident word alignment = TRUE
- source-side string has punctuation at beginning or end of span = FALSE
- target-side string has punctuation at beginning or end of span = FALSE
- begin and end words on both sides have links leading to well-formedness = FALSE

The aforementioned list can be clarified by looking at the example in the figure, with the NP and VP as the subtree roots:

Figure 3: An example of a tree pair having the feature values as described in the example list. The tree pair is the Dutch NP which is erroneously linked to the English VP. A remove rule reacting to the current feature values as triggering environment could lead to the current link being removed, while a later addition rule could add the correct link between the two NPs. For simplicity, only the relevant non-terminal links are shown.



- Clearly, these nodes do not have unary children, with or without taking punctuation into consideration. Therefore, all relevant values are 0.
- There are no confident word alignments going outside (value 0). However, between the subtrees, two fuzzy word alignments go outside: the ones linking *Bring* to *Geef* and *me*. The word *Bring* is a descendent of the English VP, but *Geef* and *me* are not descendents of the Dutch NP, meaning that the tree pair is not wellformed.
- There are less than 3 fuzzy word alignments going outside, so this value is 1.
- The leaf ratio similarity score is not less than 40 if the penalty score denominator is 80. In the example, the leaf ratio is 2/3. Using the formula presented in Kotzé et al. (2012), we subtract the absolute difference (1) divided by the denominator (80) from the leaf ratio, giving us a value of roughly 0.65. The feature value is therefore 0.
- The source tree height is 1, while the target tree height is 2. The difference is 1, which is less than 4. Therefore, the feature value is 1.
- There exists a verb on only one side, in the form of *Bring*. (value=1)
- It is not the case that a pronoun exists on only one side. (value=0)
- The trees share at least one confident word alignment, in the form of *de–the* and *schep–shovel*. (value=1)
- Neither the source nor the target trees have punctuation at either the beginning or at the end of their spans. The respective values are therefore 0.
- The beginning and end words on both sides do not all have links leading to well-formedness. The word *Bring* has two links that lead to non-wellformedness, since the words to which it links (*Geef* and *me*) are not dominated by either of the trees. (value=0)

During training, we count all occurrences of specific feature value sets in the automatic data set in order to compare them to the gold standard. We assign numbers to each unique feature profile and count whether or not the action ADD (in the case of unlinked node pairs) or REMOVE (in the case of linked node pairs) compares favourably with their equivalents in the gold standard, for every such feature profile.

For example, if there are 20 node pairs in the automatic set having the same set of binary values 00011011000111010, with 12 of them unlinked and 8 linked, then the rule "if 00011011000111010 then ADD" can be applied 12 times and the rule "if 00011011000111010 then REMOVE" can be applied 8 times.

Since the gold standard has the same trees, it follows that the same node pairs exist as in the automatic data set. Some of them may be linked and some not. Transformation rule "if 00011011000111010 then ADD" are applicable 12 times to the automatic set. If we look at the same node pairs in the gold standard, we might notice that only 8 of them are linked, meaning that 8 rule applications are right and 4 are wrong. Similarly, transformation rule "if 00011011000111010 then REMOVE" can be applied 8 times to the automatic set, but we might notice that only 2 of the relevant node pairs in the gold standard are unlinked, meaning that 2 rule applications are right and 6 are wrong.

We proceed to discover possible rules and compare them to the gold standard until we have a complete list, each having a count of matches with their occurrences in the gold standard as well as mismatches. We proceed to sort them in descending order of the number of mismatches deducted from the number of matches.

In the iteration, the best rule found is added to the list of transformations and applied to the automatic data set to create a new one, from which we newly extract features and discover rules.

This continues until no rule can be found where there is a positive difference between matches and mismatches.

5.1 Rule cutoff selection and testing

Experiments have indicated that this approach can easily lead to overtraining. To counter this, we introduce a phase where the learned transformation rules are applied to a held-out set which has its own gold standard. We then select only the rules up until the one which, when applied, leads to the highest F-score when evaluated against the gold standard. This subset of rules will then be used for testing and/or aligning/correcting new data.

For example, 80 rules might have been learned. On the cutoff set, however, application of rule 60 has the highest F-score. We therefore select only the set up to and including rule 60. The order of the rules stay the same, but selecting a cutoff rule means that not all of them will eventually be applied to new data.

It may happen that with the cutoff rule, the next few rules lead to no change. In that case, we include them as well, since together with the training data set they still lead to a net improvement.

One may wish to apply the newly selected list of rules to a test set for evaluation purposes. Obviously, a gold standard for this set is needed as well. Instead of a test set, one may apply a development test set which can be used for various experiments, so that the test set is only used for the final evaluation process.

5.2 Processing new data

Any set of learned rules can be applied to new, unprocessed data. It is important to take care that the input set has been processed to the same degree as the training, cutoff and test sets. For example, if one uses TBL to improve the output of an existing tree aligner, all automatic data sets, including the new one, must ideally contain the output of that tree aligner. Of course, it would be even better if all these sets were produced by the same method or model.

The current implementation is too slow for sequential processing of large data sets such as Europarl (proceedings of the European Parliament). However, we have adapted the system for use on our local high performance computing (HPC) cluster so that many alignment sets can be processed at once. For the sentence aligned Europarl 3 corpus, we have split the sentence pairs up into 650 files containing roughly 3000 sentence pairs each. Using TBL, it took less than two days for the entire parallel corpus to be aligned on the cluster.

6. Feature selection

While developing and testing the system on our data (see section 7), we have learned a few interesting properties of the system. We have found it useful to construct a few complex rules by implementing a relatively small combination of features in order to limit the time for training and testing. This deviates from the approach of Brill (1995) who applies a large set of simple rules to the part-of-speech tagging problem. Nevertheless, our approach has some advantages and disadvantages. Some advantages are:

- The system runs more quickly because of the limited number of features.
- It is quite powerful and high scores are reached very quickly, since all feature values are combined, creating complex rules.

Some disadvantages are:

- Unlike simple rule systems, our system overtrains easily. Since the rules are complex, it is conceivable that some of them can only be applicable on the currently investigated data set.

This can result in significantly lower scores in the testing phase on sets that are different from the current one.

- Feature selection must be done carefully in order to keep the number of features to a minimum.

We attempt to lessen the impact of the disadvantages by doing the following:

- Include a variety of sentence pairs from different domains in the training data set.
- Concentrate on the most significant features and only include these.
- Make use of a held-out set in order to select a cutoff point to minimise overtraining.

In our experiments, longer feature lists always led to overtraining issues, even when using a cutoff point. Therefore, our lists tended to be small, so we experimented with including and excluding various different features.

Some of the feature lists led to output with a high measure of precision, but had a relatively low coverage, such as checking whether a number exists on only one side, which usually means that the subtrees should be unlinked. Our data sets do not contain many numbers, so including this feature did not lead to much overall improvement. Therefore, this was not included in our final best list.

We also experimented with using different initial state annotators, which we will describe in the next section in more detail.

The following is the list of features and their associated values that led to the best overall evaluation scores:

- source node has a unary child
- target node has a unary child
- source node has a unary child, ignoring children with punctuation
- target node has a unary child, ignoring children with punctuation
- less than 1 confident (good) word alignment(s) going outside
- less than 3 less confident (fuzzy) word alignment(s) going outside
- less than 1 non-terminal alignment(s) going outside
- less than 1 confident (good) word alignment(s) going outside, ignoring punctuation
- less than 3 less confident (fuzzy) word alignment(s) going outside, ignoring punctuation
- leaf ratio similarity score is less than 40; penalty score denominator is 80
- height difference is less than 4
- verb exists on only one side
- pronoun exists on only one side
- subtrees share at least one confident word alignment
- source-side string has punctuation at beginning or end of span
- target-side string has punctuation at beginning or end of span
- begin and end words on both sides have links leading to well-formedness

7. Experiments

7.1 Data

For our experiments, we make use of data sets that are part of large parallel corpora which are freely distributed online. In the Parse and Corpus-Based Machine Translation (PaCo-MT) project (see acknowledgments), we have parsed and aligned these corpora to produce large parallel treebanks in four language directions: Dutch to English, English to Dutch, Dutch to French, and French to Dutch.

Our experimental data includes processed text from the following corpora:

- Europarl 3 (Koehn 2005), a corpus containing the proceedings of the European Parliament in several languages from 1996 to 2006.
- the DGT (Directorate-General for Translation) Multilingual Translation Memory of the Acquis Communautaire: DGT-TM.[2]
- OPUS: the Open Parallel corpUS (Tiedemann 2009).[3]
- A set of private translation memories obtained from the OneLiner company in Sint-Niklaas, Belgium.

All data sets were preprocessed by being sentence aligned, parsed, word aligned and finally tree aligned. In this work, we report the tools that we have used for processing Dutch to English.

- Sentence alignment was done by Hunalign (Varga et al. 2005). From the output, we selected only the 1:1 sentences for further processing.
- For parsing Dutch, we have used the dependency parser Alpino (van Noord 2006). For our experiments, the XML was converted to a phrase-structure format, TIGER-XML, which allows non-projectivity and imports metadata including dependencies. Since Alpino-XML contains unique identifiers referring to constituents, which are also presented in the conversion, we can use the produced alignment files to refer to the Alpino-XML treebanks, meaning no loss of information for further processing by syntax-based MT systems.
- We have parsed English with the Stanford parser using factored models (Klein and Manning 2003). Output was converted to TIGER-XML as above.
- Word alignment of the bitexts was done using GIZA++ (Och and Ney 2003) using refinement heuristics implemented by Moses (Koehn et al. 2007).
- Lingua-Align was used for tree alignment, while implementing various manual rule-based heuristics at later stages.

An important step was the word alignment process. We have found that a combination of high precision and high recall alignments provide much contextual information to help make decisions regarding the alignment of non-terminals. The *intersection* heuristics provided a low recall but high precision set of alignments, while the *grow-diag* heuristics added a specified set of neighbouring alignments with a lower confidence (Och and Ney 2003). Our opinion was that *grow-diag* was a suitably balanced choice which increased the recall somewhat but still kept the precision reasonably high.

From the processed data, we constructed various alignment sets, all of them with Dutch as the source and English as the target language, using the Stockholm TreeAligner. We have kept the automatic word alignments intact, as we seek to emulate input conditions as they would occur when applying our models. In addition, this saved us a significant amount of time. In constructing our training data, we have followed these guidelines:

- We took the previously defined well-formedness constraints into account. However, we also made alignment decisions based on real-world knowledge of concepts to which the text refers. Additionally, we considered it important whether or not it would make sense to translate the source-side string into the target-side string in at least some contexts. We have applied the constraint involving one-to-one links only to those between non-terminals. Many-to-one and even many-to-many links between terminals occur frequently in cases where there is otherwise

2. http://langtech.jrc.it/DGT-TM.html
3. http://opus.lingfil.uu.se

English phrase	**Dutch phrase**	**Translation**	**Reason**
its activities	de activiteiten die daar plaatsvinden	the activities taking place there	loose translation
to join us	tot de Unie toetreden	to join the Union	loose but correct in context
the European Union	Europese Unie	European Union	incomplete but plausible translation
the story	de verhalen	the stories	loose translation
create opinion	onze opinies vormen	create our opinion	more general translation

Table 1: Some phrases which have been linked using the fuzzy (less confident) link. Note that the translation direction is from Dutch to English, which makes the last example a more general translation instead of a more specific one.

no satisfactory way to represent translational equivalence. Because of this and because we prefer high recall over high precision alignments (see discussion on previous extrinsic evaluations in section 9), we have allowed this.

- As mentioned before, we distinguish between so-called *good* and *fuzzy* links, which denote confident and less confident links respectively.
- A good link is made between subtrees where the string dominated by the target subtree would be considered a good translation in at least some contexts of the string dominated by the source subtree.
- A fuzzy link is made between subtrees where the string dominated by the target subtree would be considered a conceivable but less optimal translation of the string dominated by the source subtree. Examples would be where the target string (*the vehicle*) is more general than the source string (*the car*). A more general translation is still in a sense correct, but a more specific translation can easily be wrong, since the knowledge about the specifics is not necessarily present on the source-side. However, in cases where the target-side is more specific and this particular translation seems to be relatively frequent, we still align them using a fuzzy link. We also make fuzzy links, for example, where the translation is loose or where less important words, such as determiners, are left out on one side.

Figure 1 gives a good example of a subtree pair which should be confidently aligned. The phrases *zwaarwichtig en onrijp* and *ponderous and immature* are word-for-word translations where every terminal has a good link to a terminal governed by the other subtree. Conversely, there may be a number of reasons to prefer aligning subtrees using a fuzzy link. Generally, though, they are pairs which we have deemed as translationally less optimal, but acceptable. Table 1 presents such examples of phrases which we have deemed to align using fuzzy links.

As mentioned in the previous section, there are a number of different data sets that can be used in the experimental process, each having its own gold standard. We therefore proceeded to construct four gold standard sets from Lingua-Align output which was trained on a set of 140 parallel sentences from Europarl which we used in the PaCo-MT project. The sets consist of:

- A training data set comprising 250 sentence pairs from four different domains. The respective average source and target sentence lengths are 20.24 and 20.6 tokens.

Function	Domain(s)	Sentence pairs
Training set	EP (50), OPUS (150), DGT (50)	250
Cutoff set	EP (50), OPUS (35), DGT (15)	100
Development test set	EP (50), OPUS (150), DGT (50)	250
Test set	EP	200

Table 2: Summarisation of the data used in the experiments, where EP = Europarl.

- A held-out data set comprising 100 sentence pairs from the same four different domains, which we use to determine a rule cutoff point, as discussed in the previous section. (21.26 / 20.24 tokens per sentence)
- A development test set comprising 250 sentence pairs from the same four different domains. (19.42 / 19.45 tokens per sentence)
- A test set comprising 200 sentence pairs from one domain (Europarl). This was done on purpose in order to measure performance on a single domain. Future test sets may be comprised from other domains. (21.86 / 20.38 tokens per sentence)

Table 2 summarises the above.

For the training and the development test set, the OPUS data is subdivided into OpenSubtitles (movie subtitles, 100 sentence pairs) and EMEA (documentation from the European Medicines Agency, 50 sentence pairs). In the cutoff set, the OPUS data are subdivided into 20 OpenSubtitles and 15 EMEA sentence pairs. We have aligned more sentences in OpenSubtitles since most of them are very short, although we have tried to include enough variation and negative examples as well (for example where parses are such that many constituents cannot be aligned).

Since all four data sets have both a corrected (gold) and an uncorrected version, this brings the total up to eight sets.

7.2 Initial state annotators

We have also experimented using different combinations of alignment steps. One alignment step is a bottom-up algorithm which we describe in Kotzé et al. (2012), which mainly uses the similarity features using geometric averages as described in the previous section. It can be described as follows:

- For every unlinked source-side non-terminal node, starting at the lowest level:
  - Inspect every unlinked target-side non-terminal node, starting at the first level.
  - If conditions hold between the current two subtrees, link them, else continue to the next unlinked target-side non-terminal node on the same level.
  - If no target-side non-terminal node has been linked on the current level, continue to the next level, and repeat the previous steps up to and including the highest possible level.
  - If the current source-side node has been linked, move on to the next source-side node on the same level, and start inspecting target-side nodes on the first level again.
  - If all source-side non-terminal nodes have been inspected on the current level, move on to the next level and repeat by inspecting unlinked target-side nodes on the first level, repeating all previous steps.
- Stop when all unlinked source-side non-terminal nodes have been processed.

This is a general algorithm which can be applied using any set of conditions. We have experimented with the similarity features as described in section 4, eventually using a combination of the leaf ratio similarity and linked leaf ratio similarity scores using specific values suited for our data set.

The algorithm can also be applied in the other direction by starting from the target nodes instead of the source nodes. One can create different data sets by applying both directions to an existing data set. Furthermore, one can create a higher precision set of the latter two sets by calculating and applying their intersection, or a higher recall set by doing the same with the union.

In our experiments, using the union set as input somehow always led to many more rules being learned, with relatively little increase in performance. We have therefore continued to experiment with these combinations (all sets were already word aligned):

- TBL
- bottom up + intersection + TBL
- Lingua-Align + TBL
- Lingua-Align + bottom up + intersection + TBL

This Lingua-Align model is the same one as mentioned before: a high precision but low recall model trained on a separate Europarl set, which we applied to all corpora in the PaCo-MT project. We have subsequently applied the source-to-target version of the bottom-up algorithm to it, which increased recall as well as the BLEU score, as reported in Kotzé et al. (2012) and Vandeghinste et al. (In press.). The fact that experiments with intersection data sets increased F-scores even further suggested that it would be a good idea to experiment with alignment in that order.

8. Results

We would like to point out that for our evaluation scores, we do not distinguish between good and fuzzy links, since we do not yet attempt to make that distinction in our output. Therefore, we use the standard balanced F-measure as it occurs in many other NLP publications, instead of the more refined definition as is used in the word alignment literature (Och and Ney 2003), where the recall of only good links is taken into consideration.

Table 3 displays our test results on the development test set using all evaluated combinations, while table 4 displays the same results on our Europarl test set. We do not include Lingua-Align scores here, because in order to be fair, we would also need to train a Lingua-Align model on the same training data set, which comprises four different domains. However, for Lingua-Align training we would also need to specify the correct GIZA++ models, and we do not have any single model that was trained on all four these domains. If we attempt to train a Lingua-Align model on the training data set specifying any single GIZA++ model, for example the Europarl model, recall drops substantially.

Therefore, we proceed to select sentence pairs from the domain which is represented the most in all our data sets, Europarl, and implement ten-fold cross validation using both Lingua-Align and TBL on its word aligned version. We do the same with the pipeline that has led to the best scores on average: Lingua-Align + bottom-up + intersection + TBL. The set consists of 350 sentence pairs, on which we train 90% - i.e. 315 sentence pairs - and test on the rest. For Lingua-Align, we use the same set of parameters that we used in the PaCo-MT project, with an improved set of features that resulted in better scores for later experiments. For TBL, we use the same set of template instantiations that led to the scores above. We present the results of our evaluation in Table 5.

A number of things stand out when looking at our results. First of all, applying the intersection of the bottom-up algorithm generally leads to a high precision but a low recall.

Alignment steps	**Precision**	**Recall**	**F-score**
Intersect	**91.2**	47.6	62.6
TBL	88.3	78.3	83.0
Intersect+TBL	87.4	81.2	84.2
Lingua-Align+TBL	89.0	80.0	84.3
Lingua-Align+intersect+TBL	87.8	**82.7**	**85.2**

Table 3: Evaluation results on the development test set

Alignment steps	**Precision**	**Recall**	**F-score**
Intersect	**79.3**	51.8	62.7
TBL	78.2	79.3	78.7
Intersect+TBL	77.4	82.5	79.9
Lingua-Align+intersect+TBL	77.3	**83.9**	**80.5**

Table 4: Evaluation results on the Europarl test set

Secondly, all TBL experiments lead to very good results on the development test set and worse results on the test set. This is expected since the rules were learned while training on a set with similar data than the development test set, whereas the test set consists of only Europarl data. Generally, the combination consisting of Lingua-Align, the bottom-up intersection and TBL fares the best although the F-score of Lingua-Align+TBL is a 0.1 improvement on the test set. The scores of Lingua-Align+intersect+TBL and Lingua-Align+TBL are generally very close to each other, suggesting that the combination of Lingua-Align and TBL is more important than using the intersection with TBL.

However, the score from the Lingua-Align model with the intersection without using TBL is also quite high for the test set, as Table 6 indicates. Performance is inconsistent, however, as the F-score for the same combination on the development test set is much lower. Performance of Lingua-Align varies greatly between the sets as well. This is not surprising as the Lingua-Align model was trained on sentences from Europarl. However, note that Lingua-Align precision stays very high throughout.

Thirdly, the TBL experiments produce very high precision results on the development test set with lower but still good recall scores. But when applied to the test set, recall is always as least as good as precision. The differences are not that much but still noticeable. We do not yet have a clear explanation for this, but we speculate that the generalizing nature of the algorithm, together with the use of a cutoff data set, may be at least partly the reason for the general high recall scores.

Finally, it is clear that our ten-fold cross validation results favour TBL over Lingua-Align. The very high scores of the error correcting model (Lingua-Align+bottom-up+intersection+TBL) are consistent with those of the development test set and the test set. As a tree aligner, TBL scores surprisingly high using only a very small set of features.

Figure 4 is a graphical presentation comparing the test set evaluation scores for the TBL experiments that we reported in Tables 3 and 4 as more rules are applied for each set. Note that the plots

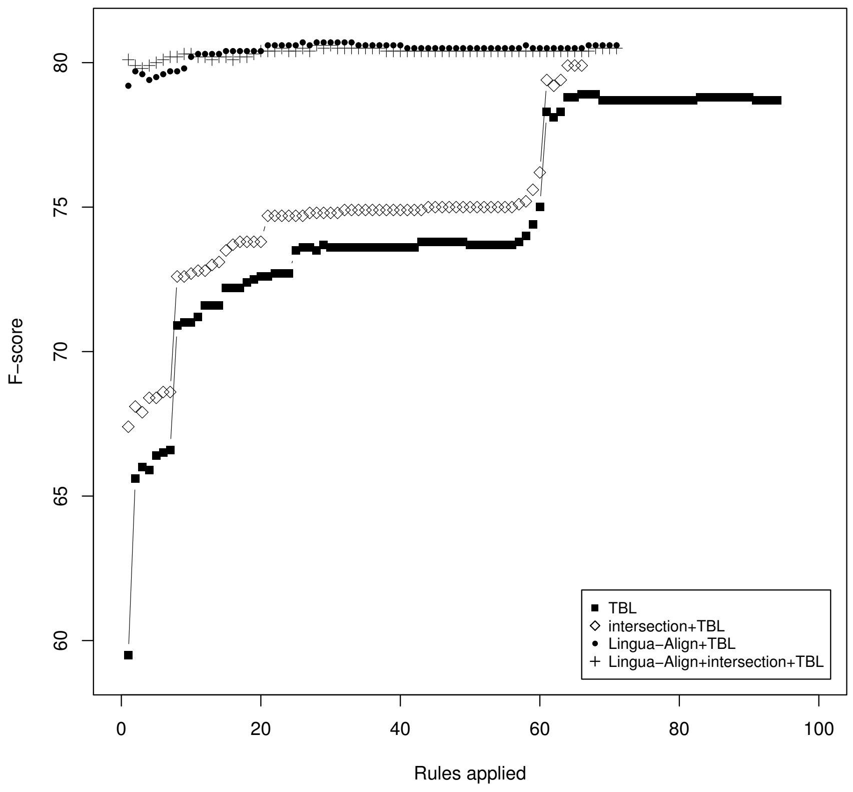
System / Pipeline	**Precision**	**Recall**	**F-score**
Lingua-Align	81	77.6	79
TBL	**84.9**	80.1	82.5
Lingua-Align+bottom-up+intersect+TBL	84.2	**84.2**	**84.2**

Table 5: Ten-fold cross validation scores applied on Europarl data for TBL and Lingua-Align

System / Pipeline	Data set	Precision	Recall	F-score
Lingua-Align	devtest set	**91.3**	41.6	57.2
Lingua-Align+intersect	devtest set	85.1	57.5	68.6
Lingua-Align	test set	90.1	60.5	72.4
Lingua-Align+intersect	test set	81.3	**76.9**	**79.0**

Table 6: Evaluation scores of Lingua-Align with and without adding the intersection of the bottom-up algorithm, on both the test and development test sets.

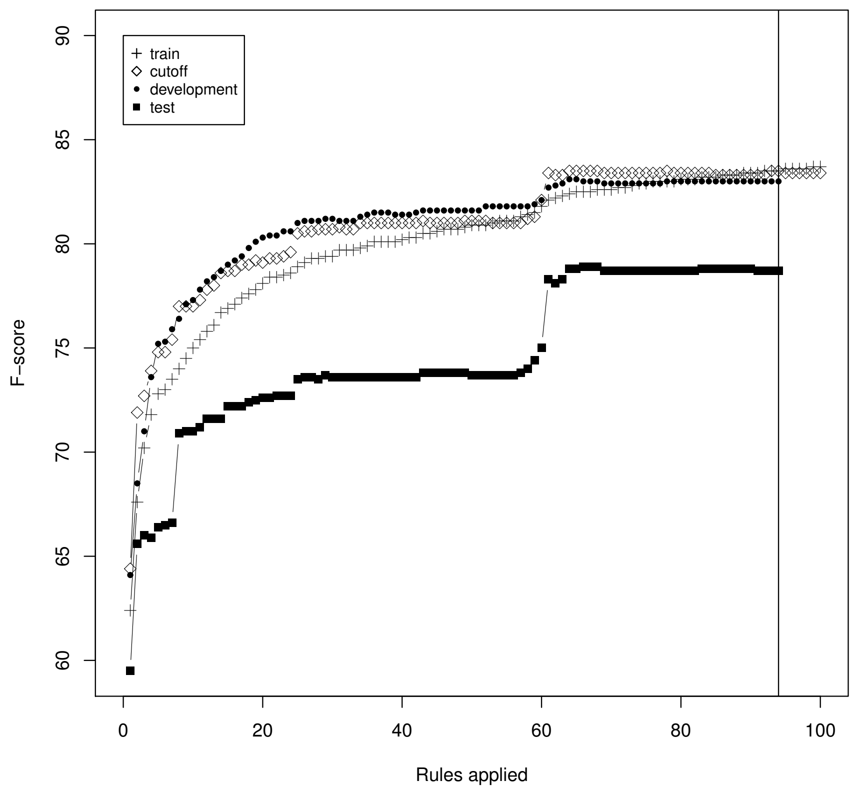
Figure 4: Comparison of the F-scores of the four different test sets.



are of different lengths because of the fact that a different number of rules was applied in each case. It is clear that the first few rules are very effective in the case of TBL and intersection+TBL. Unlike with training, where the points are usually plotted in a shape resembling a logarithmic curve, with these two experiments, there are three distinct jumps in performance increase at more or less the same points on the x axis. Note that we do not include the original scores in the graph (at x=0).

In the case of Lingua-Align+TBL and Lingua-Align+intersection+TBL, they start off with a high initial score and increase only slightly after application of the rules. Their great similarity shows that applying the intersection algorithm to the Lingua-Align data set is sufficient to produce a high score, as is also clear from table 6. However, this is only true for the test set, since the difference

Figure 5: Comparison of the F-scores of the four different evaluation sets when the system is used as a tree aligner.



in scores between Lingua-Align+intersect and Lingua-Align+intersect+TBL is quite large for the development test set.

Figure 5 gives us some insight to what happens when we use the system as a tree aligner. We plot the F-scores for not only the test set, but also for the training, cutoff and development test sets. The vertical line shows where the cutoff point is, which is the maximum F-score obtained in the cutoff data set.

It is clear that the scores for the other sets are not only higher than those of the test set, but that they also more closely resemble a logarithmic function, the closest of which is displayed by that of the training data set. However, the points of the cutoff and development sets mimic the last jump in F-score improvement in the test set graph to a degree. It is also striking that after only a single rule application, F-scores jump to between 59.5 and 64.4. During training, the first rule adds 910 alignments of which only 132 are incorrect. However, test set results are consistently lower, which suggests overtraining, in spite of the measures undertaken to minimize its effect.

The first few rules applied are very similar, even when applied to the different experimental data sets. Most rules add alignments and significantly fewer remove them. This obviously holds more strongly when TBL is applied to a set containing just word alignments. With all four experiments, the first rule learned is the same. These are the features associated with this rule:

- The source and target roots are not in a unary relation, even when accepting unary parents that have punctuation as children (see section 4.1.1).
- No more than two fuzzy word alignments exist that lead to non-wellformedness (see section 4.1.2).
- No more than two fuzzy word alignments exist that lead to non-wellformedness, ignoring punctuation.
- No good word alignments exist that lead to non-wellformedness.
- No good word alignments exist that lead to non-wellformedness, ignoring punctuation.
- No non-terminal alignments exist that lead to non-wellformedness.
- The leaf ratio score is 40 or more, with a penalty score denominator of 80. Recall that the higher the leaf ratio score, the higher the similarity. However, we use a penalty score to take the leaf count difference into account, using a normalization value to avoid subtracting a too large amount.
- The height difference is less than 4 (see section 4.1.3).
- A verb is either present on both the source and target-side, or none (see section 4.4.1).
- A pronoun is either present on both the source and target-side, or none.
- The subtrees share at least one good word alignment (see section 4.2.2).
- The first and last terminal of each subtree has a link that goes to a descendant terminal of the root of the other subtree ("inside"). In other words, these four terminals - the start and end terminals of the source and target-sides - all link to terminals dominated by the subtrees with the current source and target non-terminals as roots (see section 4.4.2).

Figure 6 displays an example of a set of trees conforming to the requirements set by the first rule.

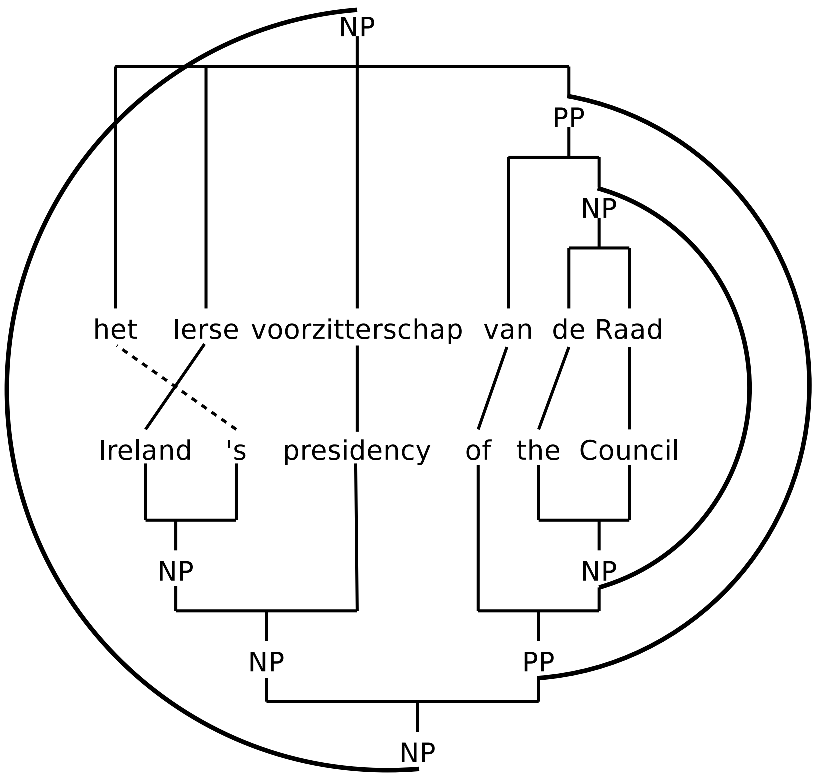
In all cases, the second rule is almost exactly the same. Subsequent rules follow similar patterns, allowing for certain conditions to be true where it was not the case in the first rule, for example that a verb is present on only one side. With all four experimental sets, the first REMOVE rule (situated at position 4, 5 or 6) is exactly the same. This rule also only differ with only one respect to the first ADD rule which is the removal of links where there is at least one non-terminal alignment leading to non-wellformedness. The fact that they do not differ that much makes sense, because we remove alignments that have been added before, meaning that they share a significant number of features that are also good for adding. REMOVE rules also noticeably become more frequent towards the end, but the scope of their application also drastically diminishes.

8.1 Constructing the parallel treebanks

After completing all our experiments, we proceeded to apply the rules on the full versions of the parallel treebanks mentioned in section 7 which were used in the PaCo-MT project. On average, Lingua-Align+intersection+TBL has led to the best results. Therefore, we first produced two versions using the bottom-up algorithm on the already existing Lingua-Align treebanks and then produced a third version using the intersection. Finally, we apply the rules that we learned for this experimental data set up to and including the cutoff point that was determined by the held-out set.

Table 7 displays some count statistics after the complete alignment process. Note that applying TBL removes some alignments created by Lingua-Align and the bottom-up algorithm. The difference

Figure 6: An example of a set of subtrees linked by application of the first rule. Solid lines are confident (good), the dashed line is less confident (fuzzy).



Data set	**Alignment count**
Word alignments	48,003,332
NT alignments	16,784,348
TBL alignments	4,207,679
Intersection NT alignments	5,597,165
Intersection NT alignments before TBL	5,666,905
Lingua-Align alignments after TBL	6,979,504
Lingua-Align alignments after intersection	7,346,180
Original Lingua-Align alignments	7,931,883

Table 7: Alignment counts for parallel corpora processed by Lingua-Align+intersect+TBL, where NT=non-terminal

between the Lingua-Align alignments before applying intersection and after can be explained by the fact that the bottom-up algorithm also replaces fuzzy sentence root links with good ones. It is also very clear that both the intersection of the bottom-up algorithm and TBL add a significant number of alignments, constituting roughly 40% more than the existing number of Lingua-Align alignments.

9. Future work and conclusions

We have presented a system which adapts the powerful transformation-based learning algorithm to the problem of tree-to-tree alignment. In this work, we have described a set of experiments that were run on various data sets. The evaluation scores are quite high, even as we used a relatively short list of features.

We must stress that it might be possible to train a better Lingua-Align system on the same data using the right feature and parameter sets. Nevertheless, to the best of our knowledge, the results of our TBL experiments rival and in some cases surpass the best scores that were produced by Lingua-Align. At this point in time, it is premature to compare our system to other aligners such as the Dublin aligner, which has been evaluated on different data sets. However, a series of ten-fold cross validation experiments using the same alignment sets and word alignment files as used for Table 5 would provide more clarity on the strength of its performance.

We have gone to reasonable lengths to ensure that experimental data sets contain a variety of parsed sentences from different domains. Based on the good results discussed above, we are reasonably confident that we have a working system with the ability to produce quality alignments for Dutch to English, which should also be fairly easily adaptable for other language pairs.

Our next step is to train a PaCo-MT model using our newly produced alignments and compare it to the performance of statistical systems trained on the same corpora. Table 8 displays some evaluation results of the system trained on previous models (Kotzé et al. 2012). The high precision model is trained on alignments produced by Lingua-Align. The high recall model, which produced better results, is trained on the output of a version of the bottom-up algorithm (source to target). Both alignment models have inferior results when compared to those produced by the system described in this work. We hope that in the light of this, we would be able to train an MT system based on our alignments that offers a viable alternative to the state-of-the-art statistical systems available today.

Currently, our system can still be better optimized for faster experimentation. Every iteration involves much input and output in the form of writing statistics to files and feature extraction. Various possibilities exist to optimize this step.

Our system also does not currently generate template instantiations. This means that there is much more manual work to be done when experimenting. Methods to achieve this without significant sacrifice of performance include random sampling using Monte Carlo methods (Samuel 1998) and making more independence assumptions (Ngai and Florian 2001).

Currently, our system does not add links with differing degrees of confidence, which may play a significant role in the performance of statistical MT systems.

Obviously, experimenting with additional features may improve performance in general. For example, classifying links that lead to non-wellformedness may be useful. A verb being linked this way - in other words, where the link goes outside - may be regarded as a more negative feature than a preposition, for example.

An example of a feature that can be extended is the one which checks for the existence of good word alignments between the subtrees. Instead, shared non-terminal alignments can be included, with the option of not only allowing confident alignments.

We could also incorporate GIZA++ word alignment probability features which are used by Lingua-Align and the Dublin aligner. Lingua-Align also makes use of features including looking at the position of the nodes with respect to the whole tree span, normalized instead of absolute height differences, more general category label and part-of-speech combinations, parents, grandparents, children, grandchildren, sisters, edges, and so on.

At some point, we would be interested to create separate templates for improving word alignments, which we can use in conjunction with those that we use for non-terminal alignments. The idea is to extend the system to one that can be applied to general alignment correction in a single step. Tree correction for improved alignment convergence is an additional interesting idea that could be implemented, as described in Burkett and Klein (2012).

Acknowledgement

Much of the work described in this paper was carried out within the PaCo-MT project. The PaCo-MT project was carried out within the STEVIN programme which is funded by the Dutch and Flemish Governments (http://www.stevin-tst.org).

References

Ayan, N.F., B.J. Dorr, and C. Monz (2005), Alignment link projection using transformation-based learning, *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 185–192.

Black, W.J. and A. Vasilakopoulos (2002), Language independent named entity classification by modified transformation-based learning and by decision tree induction, *Proceedings of CONLL2002*, pp. 159–162.

Brill, E. (1992), A simple rule-based part of speech tagger, *Proceedings, Third Conference on Applied Natural Language Processing, ACL*, Trento, Italy.

Table 8: Evaluation of adding data to the model

Treebanks	**High Precision**			**High Recall**		
	BLEU	NIST	TER	BLEU	NIST	TER
Europarl	22.28	6.43	64.14	25.48	7.36	61.12
Europarl+OPUS	22.71	6.57	64.68	26.23	7.40	61.63
Europarl+OPUS+DGT-TM	22.83	6.66	65.26	24.10	6.59	64.08
Europarl+OPUS+DGT-TM+Transmem	24.19	6.78	64.31	26.57	6.95	62.46
Europarl+OPUS+Transmem				**29.12**	**7.68**	**60.04**

Brill, E. (1993a), Automatic grammar induction and parsing free text: A transformation-based approach, *Proceedings, 31st Meeting of the Association of Computational Linguistics*, Columbus, OH, USA.

Brill, E. (1993b), Transformation-based error-driven parsing, *Proceedings, Third International Workshop on Parsing Technologies*, Tilburg, The Netherlands.

Brill, E. (1994), Some advances in rule-based part of speech tagging, *Proceedings, Twelfth National Conference on Artificial Intelligence (AAAI-94)*, Seattle, WA, USA.

Brill, E. (1995), Transformation-based error driven learning and natural language processing: A case study in part of speech tagging, *Computational Linguistics* **21(4)**, pp. 543–565.

Brill, E. (1996), *Chapter: Learning to Parse with Transformations*, Kluwer.

Brill, E. and P. Resnik (1994), A transformation-based approach to prepositional phrase attachment disambiguation, *Proceedings, Fifteenth International Conference on Computational Linguistics (COLING-1994)*, Kyoto, Japan.

Burkett, D. and D. Klein (2012), Transforming trees to improve syntactic convergence, *Proceedings of EMNLP*, Jeju Island, South Korea.

De Kok, D. (2008), *Headline generation for dutch newspaper articles through transformation-based learning*, Master's thesis, University of Groningen.

Florian, R. and G. Ngai (2001), Multidimensional transformation-based learning, *Proceedings of the 2001 workshop on Computational Natural Language Learning - Volume 7*, ConLL '01, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1:1–1:8.

Florian, R., J.C. Henderson, and G. Ngai (2000), Coaxing confidences from an old friend: probabilistic classifications from transformation rule lists, *Proceedings of the 2000 Joint SIGDAT conference on Empirical methods in natural language processing and very large corpora: held in conjunction with the 38th Annual Meeting of the Association for Computational Linguistics*, Vol. 13 of *EMNLP '00*, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 26–34.

Fung, P., G. Ngai, Y. Yang, and B. Chen (2004), A maximum-entropy chinese parser augmented by transformation-based learning, *ACM Transactions on Asian Language Information Processing (TALIP)* **3** (2), pp. 159–168, ACM, New York, NY, USA.

Groves, D., M. Hearne, and A. Way (2004), Robust sub-sentential alignment of phrase-structure trees, *Proceedings of the 20th International Conference on Computational Linguistics (COLING-2004)*, pp. 1072–1078.

Hearne, M., J. Tinsley, V. Zhechev, and A. Way (2007), Capturing translational divergences with a statistical tree-to-tree aligner, *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI)*, Skövde, Sweden.

Higgins, D. (2004), A transformation-based approach to argument labeling, *in* Ng, H.T. and E. Riloff, editors, *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, Association for Computational Linguistics, Boston, Massachusetts, USA, pp. 114–117.

Kavallieratou, E., E. Stamatatos, N. Fakotakis, and G. Kokkinakis (2000), Handwritten character segmentation using transformation-based learning, *Proceedings of International Conference on Pattern Recognition (ICPR)*, IEEE Computer Society, pp. 2634–2637.

Klein, D. and C. Manning (2003), Fast exact inference with a factored model for natural language parsing, *Advances in Neural Information Processing Systems 15 (NIPS 2002*, Cambridge, MA: MIT Press, pp. 3–10.

Koehn, P. (2005), Europarl: a parallel corpus for statistical machine translation, *Proceedings of MT Summit X*, IAMT, pp. 79–97.

Koehn, P., H. Hoang, A. Birch, C. Callison-Burch, M. Federico, N. Bertoldi, B. Cowan, W. Shen, C. Moran, R. Zens, Dyer D., O. Bojar, A. Constantin, and E. Herbst (2007), Moses: Open source toolkit for statistical machine translation, *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic, pp. 177–180.

Kotzé, G. (2011a), Improving syntactic tree alignment through rule-based error correction, *Proceedings of ESSLLI 2011 Student Session*, Ljubljana, Slovenia, pp. 122–127.

Kotzé, G. (2011b), Rule-induced error correction of aligned parallel treebanks, *Proceedings of Corpus Linguistics*, Saint Petersburg, Russia, pp. 35–40.

Kotzé, G., V. Vandeghinste, S. Martens, and J. Tiedemann (2012), Large aligned treebanks for syntax-based machine translation, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC'12)*, European Language Resources Association (ELRA), Istanbul, Turkey, pp. 467–473.

Lavie, A., A. Parlikar, and V. Ambati (2008), Syntax-driven learning of sub-sentential translation equivalents and translation rules from parsed parallel corpora, *Proc. SSST*, pp. 87–95.

Mangu, L. and E. Brill (1997), Automatic rule acquisition for spelling correction, *Proceedings of the 14th International Conference on Machine Learning*, Morgan Kaufmann, pp. 734–741.

Menezes, A. and S.D. Richardson (2001), A best-first alignment algorithm for automatic extraction of transfer mappings from bilingual corpora, *Proceedings of the ACL 2001 Workshop on Data-Driven Methods in Machine Translation*, Toulouse, France, pp. 39–46.

Ngai, G. and R. Florian (2001), Transformation-based learning in the fast lane, *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language Technologies*, NAACL '01, Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1–8.

Och, F.J. and H. Ney (2003), A systematic comparison of various statistical alignment models, *Computational Linguistics* **29** (1), pp. 19–51.

Ramshaw, L.A. and M.P. Marcus (1995), Text chunking using transformation-based learning, *CoRR* pp. 82–94.

Samuel, K. (1998), Lazy transformation-based learning, *in* Cook, Diane J., editor, *Proceedings of the Eleventh International Florida Artificial Intelligence Research Society Conference*, AAAI Press, pp. 235–239.

Samuel, K., R. Carberry, and K. Vijay-shanker (1998), Dialogue act tagging with transformation-based learning, *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pp. 1150–1156.

Tiedemann, J. (2009), News from opus - a collection of multilingual parallel corpora with tools and interfaces, *Proceedings of Recent Advances in Natural Language Processing (RANLP-2009)*, pp. 237–248.

Tiedemann, J. (2010), Lingua-align: An experimental toolbox for automatic tree-to-tree alignment, *Proceedings of the 7th International Conference on Language Resources and Evaluation (LREC'2010)*, Valetta, Malta.

van Noord, G.J. (2006), At last parsing is now operational., *Proceedings of Traitement Automatique des Langues Naturelles (TALN)*, pp. 20–42.

Vandeghinste, V., S. Martens, G. Kotzé, J. Tiedemann, J. Van den Bogaert, K. De Smet, F. Van Eynde, and G. Van Noord (In press.), Parse and corpus-based machine translation, *Essential Speech and Language Technology for Dutch: resources, tools and applications, Springer.*

Varga, D., L. Németh, P. Halácsy, A. Kornai, V. Trón, and V. Nagy (2005), Parallel corpora for medium density languages, *Proceedings of Recent Advances in Natural Language Processing (RANLP-2005)*, pp. 590–596.

Williams, K., C. Dozier, and A. McCulloh (2004), Learning transformation rules for semantic role labeling, *Proceedings of CoNLL-2004*, Association for Computational Linguistics, Boston, MA, USA, pp. 134–137.

Zhechev, V. and A. Way (2008), Automatic generation of parallel treebanks, *Proceedings of the 22nd International Conference on Computational Linguistics (COLING-2008)*, pp. 1105–1112.