Automatic Generation of Parallel Treebanks:

An Efficient Unsupervised System

Ventsislav Zhechev, B.A.

A dissertation submitted in fulfilment of the requirements for the award of

Doctor of Philosophy (Ph.D.)

to the

Dublin City University
School of Computing

Supervisor:
Prof. Andy Way

July 2009

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy (Ph.D.) is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: _____ (Ventsislav Zhechev) ID No.: 56124058

Date: 29.09.2009

Contents

ACKNOWLEDGEMENTS....1

1. INTRODUCTION....2

2. RELATED WORK....6

2.1. Parallel Treebanks....9

2.2. Sub-Tree Alignment....16

2.2.1. Dependency Structures....16

2.2.2. Phrase-Structure Trees....20

2.3. Conclusions....26

3. SYSTEM DESIGN....30

3.1. Tree-to-Tree Alignment....31

3.1.1. Translational Equivalence....34

3.1.2. Greedy-Search Algorithm....37

3.1.3. Full-Search Algorithm....43

3.2. Other Alignment Modules....46

3.3. Re-Scoring....48

3.4. Algorithm Complexity....49

3.4.1. Space Complexity....49

3.4.2. Time Complexity....52

3.5. Conclusions....56

4. EVALUATION AND TESTING....58

4.1. Intrinsic Evaluation....61

4.1.1. Analysing the Parallel Treebanks....62

4.1.2. Evaluation Against the Gold Standard....69

4.1.3. The Full-Search-Based Algorithm as a Benchmark....74

4.2. Extrinsic Evaluation....79

4.3. Conclusions....96

5. FURTHER AVENUES FOR DEVELOPMENT AND EVALUATION....98

5.1. Improvements to the algorithms....98

5.2. Further Analysis and Evaluation....101

6. CONCLUSION....104

BIBLIOGRAPHY....107

APPENDIX A. ANALYSIS DATA TABLES

APPENDIX B. EVALUATION DATA TABLES

Automatic Generation of Parallel Treebanks: An Efficient Unsupervised System

(abstract)

Ventsislav Zhechev

The need for syntactically annotated data for use in natural language processing has increased dramatically in recent years. This is true especially for parallel treebanks, of which very few exist. The ones that exist are mainly hand-crafted and too small for reliable use in data-oriented applications. In this work I introduce a novel open-source platform for the fast and robust automatic generation of parallel treebanks through sub-tree alignment, using a limited amount of external resources. The intrinsic and extrinsic evaluations that I undertook demonstrate that my system is a feasible alternative to the manual annotation of parallel treebanks. Therefore, I expect the presented platform to help boost research in the field of syntax-augmented machine translation and lead to advancements in other fields where parallel treebanks can be employed.

Acknowledgements

I would like to thank my family for their full support and patience, and especially my wife Sveta and my son Yassen who had to put up with my living thousands of kilometres away from them. I dedicate this thesis to my grandparents Darinka and Dobrin in Bulgaria, who might be the happiest of all for the completion of my PhD degree.

I would also like to thank Andy Way, Khalil Sima'an, John Tinsley and Mary Hearne from the ATTEMPT project and the remaining NCLT and CNGL researchers at Dublin City University for the fruitful discussions and active collaboration. Special thanks go to Andy Way, who recruited me for the ATTEMPT project even though I did not have any prior experience in the field of machine translation.

My appreciation goes to the members of Jim Boylan's Kenpo Karate Club in Dublin for their moral support and for giving me the opportunity to relax and forget all science twice a week.

This work would not have been possible without the generous support by Science Foundation Ireland (grant 05/RF/CMS064) and the Irish Centre for High-End Computing (http://www.ichec.ie).

1. Introduction

In recent years much effort has been made to make use of syntactic information in statistical machine translation (MT) systems (Hearne and Way, 2006, Nesson et al., 2006, Chiang, 2007, Lavie, 2008, Li et al., 2009, Venugopal and Zollmann, 2009). This has led to increased interest in the development of parallel treebanks as the source for such syntactic data. They consist of a parallel corpus (or 'bitext'), both sides of which have been parsed and aligned at the sub-tree level.

The sentences in a parallel treebank can be parsed using either phrase-structure trees or dependency structures. Theoretically, phrase-structure-based parallel treebanks may have crossing edges and traces; dependency-based parallel treebanks, may have non-projective dependency analyses. However, this work will concentrate on phrase-structure-based parallel treebanks that do not include crossing edges and traces. Such phrase-structure analyses are commonly used in MT and computationally more efficient than their counterparts that may include crossing edges and traces.

So far parallel treebanks have been created manually or semi-automatically (Čmejrek et al., 2004, Uchimoto et al., 2004, Uibo et al., 2005, Samuelsson and Volk, 2006, Megyesi et al., 2008). This has proven to be a laborious and time-consuming task that is prone to errors and inconsistencies (Samuelsson and Volk, 2007). Because of this, only a few parallel treebanks exist and none are of sufficient size for productive use in any statistical MT application (cf. section 2.1). This leads to the first research question of this work:

> **Can a system be developed that would allow for the fast and robust automatic generation of parallel treebanks?**

Here I present an open-source platform for the automatic generation of parallel treebanks from parallel corpora. I discuss algorithms both for cases in which monolingual phrase-structure parsers (Bikel, 2002, Schmid, 2004) exist for both languages and for cases in which such parsers are not available. Having developed this system, the second research question is the following:

> **Will the parallel treebanks generated using the system presented in this thesis be of sufficient quality for effective use in Machine Translation tasks?**

I answer this through thorough intrinsic and extrinsic evaluations, which show that the parallel treebanks created with the methods described in this work are of a high quality and can successfully be used as training data for data-oriented MT systems (eg. DOT – Hearne and Way, 2006). The parallel treebanks can also be used for other statistical MT applications (eg. extraction of phrase-alignment tables for phrase-based statistical MT – Tinsley et al., 2009) and for translation studies.

I start in Chapter 2 with a comprehensive survey of existing methods for sub-sentential alignment and hand-crafted parallel treebanks. This survey shows the challenges that come up during the building of a parallel treebank, as well as the drawbacks of existing automatic systems. The conclusion that we can draw from this chapter are that a new, more robust and versatile system is needed that will enable the fast automatic generation of parallel treebanks that supersede any such existing hand-crafted treebanks both in size and quality.

Next, in Chapter 3, I describe the underlying design of the system that I developed. There, I present several different configurations that the system can use, as well as the four main operation modes: *tree-to-tree*, *tree-to-string*, *string-to-tree* and *string-to-string*. Both an exhaustive full-search-based al-

gorithm and a fast, robust greedy-search-based alternative for the determination of the best set of alignments per sentence pair were developed and are described in this chapter. I also present an analysis of the time and space complexity of the proposed system.

In Chapter 4, I present the extensive evaluation and analysis that I performed on parallel treebanks produced using my system for two different data sets, namely the English–French HomeCentre and an excerpt of the English–German part of the EuroParl parallel corpus. The results that I obtained show clearly that my system is a feasible alternative to the manual annotation of parallel treebanks and I expect it to be an effective tool for further research in the field of syntax-augmented MT.

In Chapter 5, I present some possible avenues for further development and evaluation of my system and I conclude in Chapter 6.

Portions of this work have been peer-reviewed and published in the proceedings of a number of prestigious international conferences. The paper "Robust Language Pair-Independent Sub-Tree Alignment" (Tinsley et al., 2007) is the first publication of the novel algorithms for sub-sentential alignment that lay in the foundation of the current work (cf. section 3.1.1 and section 3.1.2). It also presents some very promising initial evaluation results. This paper was presented at the MT Summit XI in Copenhagen, Denmark.

In "Capturing Translational Divergences with a Statistical Tree-to-Tree Aligner" (Hearne et al., 2007) we present detailed translational analysis of a version of the HomeCentre corpus aligned using the methods described in the paper above. This analysis shows the ability of the presented algorithms to capture complex translational divergences, like one-to-many and many-to-one correspondences, nominalisations, etc. The paper was presented at the Eleventh International Conference on Theoretical and Methodological Issues in Machine Translation in Skövde, Sweden.

"Automatic Generation of Parallel Treebanks" (Zhechev and Way, 2008) presents for the first time the complete platform for the automatic generation of parallel treebanks. It discusses the *string-to-string* alignment module (cf. section 3.2) and the full-search-based selection algorithm (cf. section 3.1.3). It also presents updated evaluation results, including using the full-search-based algorithm as an evaluation metric (cf. section 4.1.3). I presented this paper at the Twenty-second International Conference on Computational Linguistics in Manchester, the UK.

Finally, with the paper "Unsupervised Generation of Parallel Treebanks through Sub-Tree Alignment" (Zhechev, 2009) I presented my system to the open-source community. This paper includes a complete presentation of the employed algorithms, as well as exhaustive usage and installation information. It was presented within the open-source track of the Third MT Marathon in Prague, the Czech Republic and was published in the Prague Bulletin of Mathematical Linguistics.

2. Related Work

I start off the exposition with a survey of existing parallel treebanks and alternative sub-tree alignment systems. First, in section 2.1, I look at several phrase-structure- and dependency-based hand-crafted parallel treebanks and discuss their usability in the Machine Translation (MT) context. The main difference between a phrase-structure tree and a dependency-based analysis of a sentence is that on the one hand, the phrase-structure tree provides an abstract syntactic structure for the sentence in a way that positions the words of the sentence as the leaves of the resulting tree (see Figure 1).

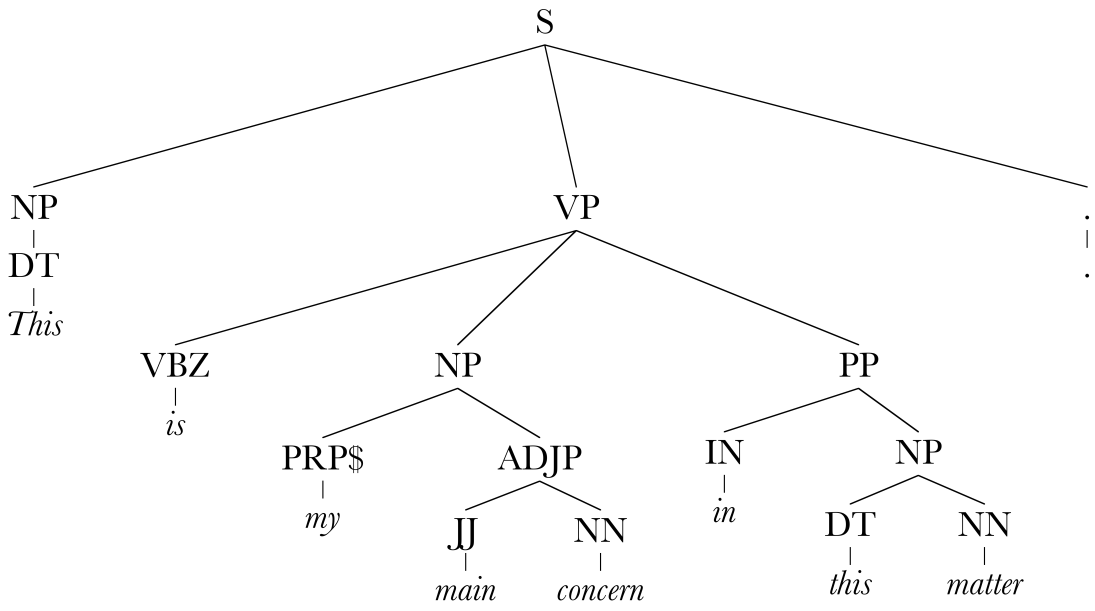


Figure 1: An example of phrase-structure analysis of a sentence

On the other hand, a dependency-based analysis of a sentence provides information about the syntactic dependencies between the words in the sentence. In Figure 2, both a traditional and a tree-based representation of the dependency-based analysis of the sentence from Figure 1 can be seen (I have omitted the syntactic categories of the dependency relations for simplicity). The latter representation is the one usually employed for sub-sentential alignment of dependency structures. As can be seen, here each

node in the tree structure is a word from the sentence that is being analysed, as opposed to the phrase-structure tree where only the leaf nodes are words and the rest are syntactic-category labels. Henceforth, when I use the term 'dependency structure' I will be referring to the analysis from Figure 2, unless explicitly stated otherwise.

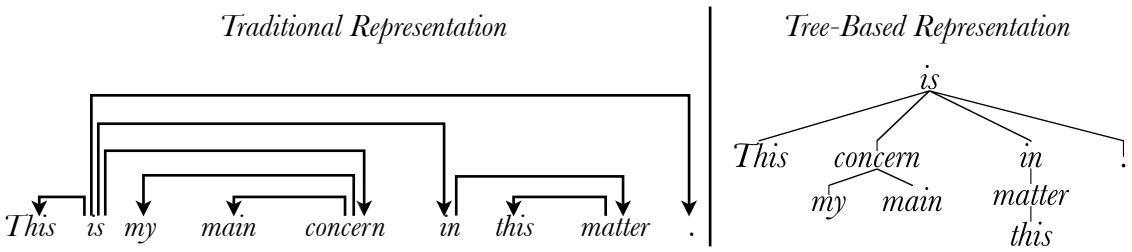


Figure 2: An example of dependency-based analyses of a sentence

Although I try to provide a complete overview of the presented parallel treebanks, I am particularly interested in several specific features they may or may not have. First, a parallel treebank may not include one-to-many structural links. There are two reasons for this. On one side, there is no agreement in the field as to whether one-to-many links should be treated in a conjunctive or in a disjunctive manner. This means that if a system using a parallel treebanks were to expect one-to-many links, it would have to be built to support both types of treatment, which increases the development complexity. On the other hand, the information encoded by conjunctive one-to-many links is implicitly present in a structural alignment scenario; while the ambiguity represented by disjoint one-to-many links makes their treatment computationally more complex.

Another important factor is the alignments present in a parallel treebank should not be based on features of the constituent-labelling schema that is used or on any language-specific features. The reasons for this are that such annotation cannot easily be adapted for other language pairs and that it might not lend itself for use by general applications employing parallel treebanks. The latter factor is also why I prefer parallel treebanks that were not built for a specific linguistic task.

Last, but not least, as I am interested in using parallel treebanks as a training resource for MT, the size of a parallel treebank is an important feature. This is especially true, given that the usual size of parallel corpora used for phrase-based statistical MT lies in the millions of sentence pairs.

In section 2.2, I discuss the development of the area of sub-tree alignment, looking at existing systems and their advantages and disadvantages compared to my system from Chapter 3. I also try to evaluate the usability of the presented methods for the automatic generation of parallel treebanks. The latter evaluation will be based on several factors that I believe are important for the automatic generation of parallel treebanks.

First, a system that automatically induces sub-structural alignments should not modify the existing structures, be it to impose isomorphism or for any other reason. I regard any existing structure as a tuned encoding of the structural information of the language in question and believe that this encoding should be preserved.

Another important point is that an automatic alignment system should be independent of the specific linguistic framework, constituent-labelling schema and language pair used. The reason for this is that any reliance on one or more of these factors will require (potentially significant) readjustment and/or redesign of the system in question. Rather, I would prefer a system that can robustly handle as many setups as possible. Also, the sentence length of the data that can be analysed should not be unnecessarily restricted.

Although any such automatic system will require the use of some bilingual dictionary or word-alignment data, it should not be over-reliant on this resource, as this can easily lead to propagation of alignment errors.

Finally, an automatic sub-tree alignment system should not be linked to a particular MT grammar-extraction task. More importantly, the system should be able to operate autonomously and produce generic parallel treebanks that do not impose restrictions on the system that will eventually use them as a training resource.

2.1. Parallel Treebanks

In this section I look at several attempts at creating parallel treebanks besides the HomeCentre treebank presented in section 4.1.

Closest to the material presented in this work comes the SMULTRON parallel treebank (Samuelsson and Volk, 2006). This manually created treebank aligns three languages—German, English and Swedish—consisting of over 1,000 sentences from each language. This comprises the first two chapters from Jostein Gaarder's novel 'Sofies verden'[1] (1991) as well as several financial texts. While Samuelsson and Volk align phrase-structure trees similarly to the methods presented in this thesis, there are subtle differences in the manner in which alignments are produced. The biggest difference is that they allow many-to-many lexical alignments and one-to-many non-lexical alignments. As will be discussed in greater detail in section 3.1, I believe such alignments to be unnecessary, because the information they carry is implicitly present in the aligned trees and could be derived, even if those alignments were not present. Furthermore, as discussed earlier, such alignments make the parallel treebanks harder for use in NLP applications.

Samuelsson and Volk also allow unary productions in the trees, which, as stated in section 3.1, do not provide any additional information useful for representing the available translational-equivalence relations between the languages in question. Additionally, Samuelsson and Volk annotate two types of alignment links: exact and approximate. For comparison, the method presented in this thesis gives the option to output the alignments together with their translational-equivalence scores, which give a far more fine-grained ranking amongst the links.

A further attempt to align phrase-structure trees is presented by Uibo et al. (2005), where the authors develop a rule-based method for aligning Estonian and German sentences. The parallel treebank consist of over 500

[1] Further in the text, I will refer to this book with its English title, 'Sophie's World.'

sentences from the novel 'Sophie's World' and in the version presented only NPs are aligned. Initially, alignment is performed by translating the Estonian and German NPs into English using electronic bilingual dictionaries and searching for intersections in the English translations. This, however, proved problematic, as often NPs in one language would correspond to other constituents in the other language; most commonly a German PP would correspond to an Estonian NP. By contrast, the system presented in my work does not suffer from such problems, as it operates independently of the constituent labelling schema used in the annotation of the syntactic structures, cf. section 3.1. As an alternative approach, Uibo et al. plan to use statistical word alignment methods instead of bilingual dictionaries for the further alignment of their parallel treebank, but that would not necessarily solve the problems arising from the inherent structural differences between Estonian and German.

Han et al. (2002) claim to have built a Korean–English parallel treebank with over 5,000 phrase-structure tree pairs from military language training manuals. The sentences seem to be relatively short, as each language side only contains about 50,000 tokens. Another property of this treebank is that the sentences are not natural utterances, but are constructed specifically for the purposes of language training. The phrase-structure representations of the English sentences follow the guidelines for the Penn Treebank (Marcus et al., 1993), while the guidelines for the parsing of the Korean sentences are presented in (Han et al., 2001). It is unclear, however, what sub-sentential alignments exactly are included in this treebank. This treebank is then referred to by Dras and Han (2002), when the authors use it to analyse the extent to which S-TAG (Shieber, 1994) can model such widely differing languages as Korean and English. Unfortunately, this paper again does not point to the type of sub-sentential alignments that might be present in the treebank. It is, therefore, uncertain whether this treebank can be used for MT applications.

Although the Prague Czech–English Dependency Treebank (PCEDT – Čmejrek et al., 2004) can be used as a parallel treebank, it is not such *per se*, in the sense that it does not directly incorporate sub-tree alignment data.

The authors do not use phrase-structure trees. Instead, tectogrammatical dependency structures are used (Hajičová, 2000), which represent the deep syntactic structure of the sentences using base forms of the words, rather than inflected forms. An example of such a structure is shown in Figure 3.

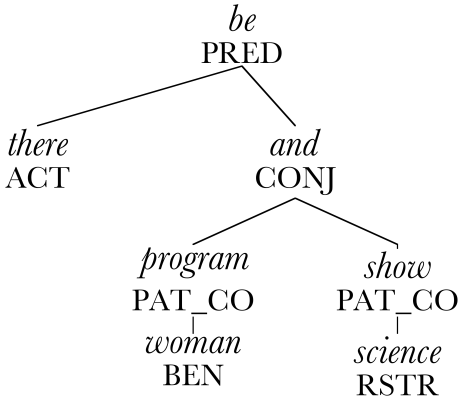


Figure 3: A tectogrammatical representation of the sentence *There's a program for women and a science show.*

Either a word-alignment tool like GIZA++ (Och and Ney, 2003) or a probabilistic electronic dictionary (supplied with the treebank) can be used to automatically align the dependency structures. The presented version contains over 21,000 sentence pairs that can be aligned. The parallel treebank was created in several steps. First, Čmejrek et al. manually translated sentences from the WSJ section of the Penn Treebank from English into Czech. Then, the English parses from the Penn Treebank were automatically converted into what they call analytical dependency structures and consequently into tectogrammatical dependency structures. The analytical dependency structures are traditional dependency structures in the sense of in Figure 2, but they differ in that they always have an additional 'technical' root node containing the sentence ID and labelled `AuxS` and inserted above the original root node. Additionally, end-sentence punctuation is moved

under the 'technical' root node. An example of the analytical structure for the sentence form Figure 3 can be seen in Figure 4.

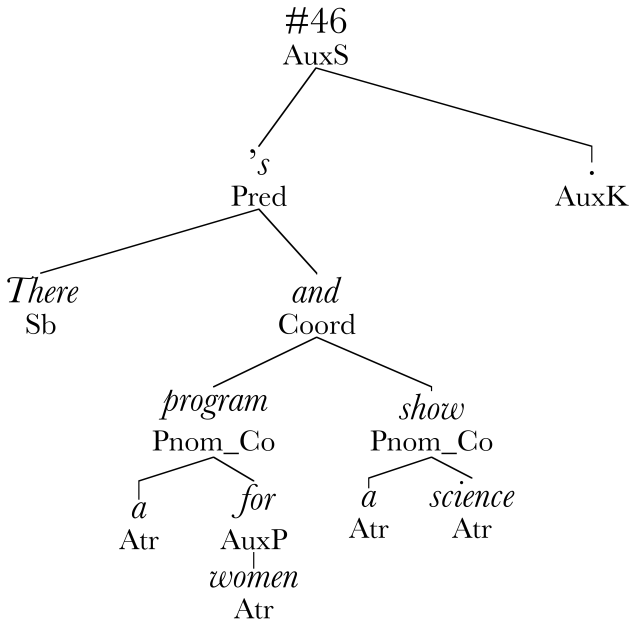


Figure 4: An analytical representation of the sentence *There's a program for women and a science show.*

The Czech sentences were automatically parsed to analytical and then converted to tectogrammatical dependency structures. Finally, Čmejrek et al. compiled probabilistic English–Czech dictionaries that can be used to find translational equivalencies in the treebank. Because of its nature, however, this treebank can only be used by MT systems that employ tectogrammatical dependency structures.

In what is currently unpublished work (Marinov, —) a Swedish–Bulgarian parallel dependency treebank is presented. It contains about 200 sentence pairs taken from the beginning of the novel 'Sophie's World.' Both sides of the parallel treebank are parsed into dependency structures automatically. The Bulgarian side was transliterated beforehand, however, rather than preserving the original Cyrillic script. The POS tags and dependency relations in the Bulgarian part were manually corrected. The sub-tree alignments are introduced manually (largely on paper, according to the author), following guidelines adapted from the guidelines for the

SMULTRON parallel treebank (Samuelsson and Volk, 2006) to be applicable to dependency structures. This parallel treebank has been built mainly as a tool to investigate the plausibility of such a task and to research possible problematic cases. If it is ever published it will present an interesting methodological study in parallel dependency treebanks, different from the PCEDT (Čmejrek et al., 2004) in that it uses traditional dependency structures (viz. Figure 2) rather than tectogrammatical ones (viz. Figure 3). Even Marinov himself, however, states that the treebank will have little practical value, as both sides of it are relatively free translations of a Norwegian original, rather than proper translations of each other.

An early effort to build a Swedish–Turkish parallel dependency treebank is presented by Megyesi et al. (2008). This treebank consists of about 165,000 tokens in Swedish and 140,000 tokens in Turkish. The sentence pairs are taken from both fiction and technical documents. Both sides of the treebank are parsed using the MaltParser (Nivre et al., 2006) into dependency structures. In this early stage of development, this parallel treebank contains word- and phrase-alignment information, but no explicit sub-tree alignments.

Uchimoto et al. (2004) present what they call a "Japanese–English–Chinese aligned parallel treebank corpora of newspaper articles." An interesting property of this treebank is the syntactic structure assumed for the Japanese sentences. For the basic syntactic building blocks they choose the so-called *bunsetsus*. These are phrasal units representing the minimal linguistic units obtained by the natural segmentation of a Japanese sentence in terms of semantics or phonetics. Each *bunsetsu* consists of one or more morphemes. For the Japanese data, dependency relations are annotated between the *bunsetsus* in a sentence, with an extra node labelled `P` used as the root of the structure. Such a dependency analysis can be seen on the right side of Figure 5.

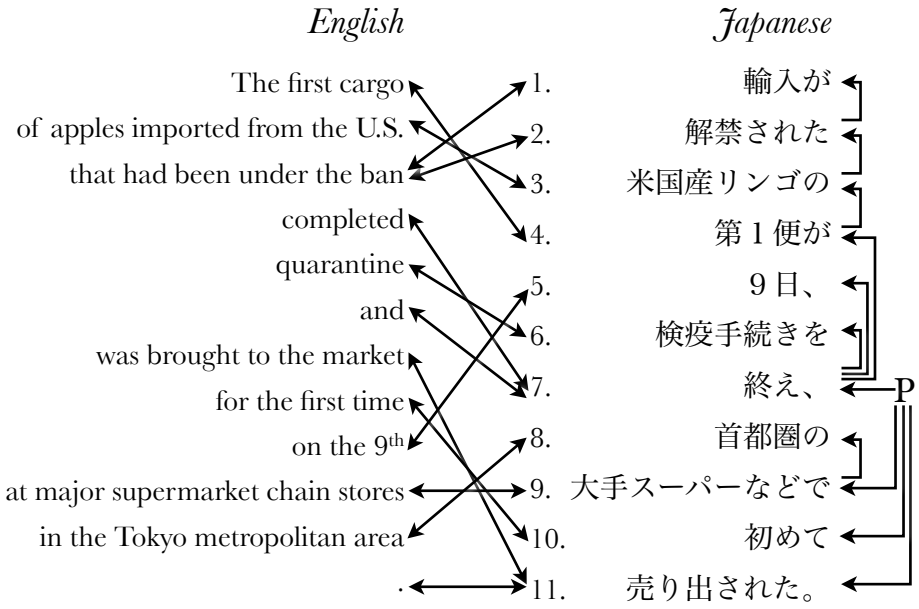


Figure 5: An example of English–Japanese alignment with dependency analysis using *bunsetsus* on the Japanese side

Uchimoto et al. then present a phrasal alignment between Japanese and English data, where in the English sentences phrases corresponding to one or more Japanese *bunsetsus* are manually identified and annotated. An example alignment is shown in Figure 5, which is an adaptation of Figure 3 from (Uchimoto et al., 2004: 66). Using these alignments, Uchimoto et al. perform a small-scale evaluation presenting the usability of the parallel treebank for example-based machine translation tasks. The treebank still has very low coverage for such tasks, however, and it is unclear how the Chinese data is aligned to the English and Japanese sides.

The Linköping English–Swedish parallel treebank (LinES – Ahrenberg, 2007) is another parallel treebank built using dependency structures. It consists of 1,200 English–Swedish sentence pairs; half are taken from online Microsoft Access help texts and half from the novel 'To Jerusalem and Back' by Saul Bellow (1976). This treebank is mainly intended as a resource for the study of translational variations and while it uses dependency structures in the sense of Figure 2, they are annotated with dependency-function labels designed specifically to serve this particular task. The word alignments (which double as sub-tree links) are automatically induced, but manually re-

viewed and adjusted to follow guidelines geared towards representing translational variations. Thus, further investigation is necessary to establish whether this parallel treebank can be used successfully for other purposes.

The CroCo corpus (Hansen-Schirra et al., 2006) consists of altogether about one million words in English–German sentence pairs in eight registers. It includes sentence pairs for which either the English or the German side is the original utterance, while its counterpart is a direct translation. Rather than being properly parsed, the sentences in the corpus are only chunked. The corpus contains word-level alignments derived using GIZA++ (Och and Ney, 2003) and chunk alignments established manually using the MMAX II tool (Müller and Strube, 2006). Each type of annotation is stored in a separate file to build a layered structure to the corpus. The main goal of Hansen-Schirra et al. is to use this corpus to investigate the assumed translation property of *explicitation*[2] (Blum-Kulka, 1986) for the language pair English–German. The corpus is designed using XML markup in such a way as to allow it to be easily queried using the XQuery language to find answers to linguistic and translational research questions. In my opinion, however, such design makes it difficult to use for MT tasks without prior transformation in a different format.

Finally, FuSe (Cyrus, 2006) contains an undisclosed number of English–German sentence pairs taken from Europarl (Koehn, 2005), such that the English sentences have been uttered by a native speaker. This is another corpus where the sentences are not parsed. Rather, predicate-argument structures are manually annotated and then aligned. Thus, in its current form this can only be used for its intended purpose: the study of translation shifts from English into German.

[2] *explicitation* is a type of semantic modification, which describes the case in which the target string (the translation) is lexically more specific than the source string (the original).

2.1.1. Sub-Tree Alignment

The approach I take to the generation of parallel treebanks is based on sub-tree alignment. Previous approaches to automatic sub-sentential alignment can be loosely grouped according to whether they focus on aligning dependency structures or phrase-structure trees. Many approaches do not view alignment as an independent task, but rather as a means to achieving another goal such as solving parse ambiguities, acquiring translation templates or bilingual grammars. Some such approaches view factors like non-isomorphism as obstacles, and alter the trees as part of the alignment process.

Other related work in the area of alignment in general views the use of tree structures as a negative aspect which may result in the loss of generalisation ability (Wellington et al., 2006). However, I choose to align predetermined tree structures without editing them; my motivation is that the structural and translational divergences that exist between source and target structures should be captured during the alignment process rather than smoothed away in order to allow for higher recall, cf. (Hearne et al., 2007). I do not view the parse trees as constraints, but rather as accurate syntactic representations of the text which can help to guide the alignment process.

2.1.2. Dependency Structures

I am particularly interested in aligning phrase-structure trees, but solutions which have been applied to the alignment of dependency structures are also relevant as they may present interesting methodological ideas for substructural alignment.

Sadler and Vendelmans (1990) present an early attempt at the automatic alignment of dependency structures. Their pilot system is example-based, language pair-dependent, semiautomatic and combines bilingual sub-sentential alignment with monolingual anaphora resolution. I would prefer, however, a fully automatic language pair-independent system that focuses on high-quality sub-tree alignment.

Matsumoto et al. (1993) present a fully automatic sub-tree alignment algorithm. The structures that they align are dependency structures similar to the ones in Figure 2, but consisting solely of the base forms of the content words from the original sentences. Additionally, eventual parse ambiguities are represented by disjunctive feature structures. Matsumoto et al. use Lexical-Functional Grammars (LFGs – Kaplan and Bresnan, 1982, Bresnan, 2001, Dalrymple, 2001) to parse English and Japanese sentences and then convert the LFG parses to the type of dependency structures that they employ. The structural matching algorithm that they present is a branch-and-bound topdown backtracking algorithm.

English: *She has long hair.*
Japanese: 彼女-の 髪-は 長い 。
she-GEN hair-TOP long .

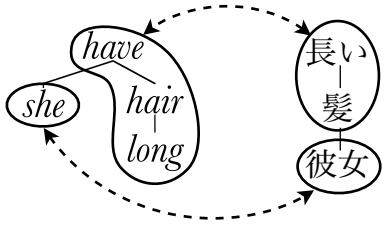


Figure 6: English–Japanese alignment of dependency *decompositions*

The actual elements that this algorithm matches, however, are not nodes in the dependency structures, but rather connected sub-graphs, which Matsumoto et al. call *decompositions*. A simple example of such an alignment is shown in Figure 6, which corresponds to Figure 1 from (Matsumoto et al., 1993: 25). Here, the encircled sub-graphs of the dependency structures are the aligned *decompositions*. During its operation, the algorithm not only discovers correspondences between *decompositions*, but also resolves existing ambiguities in the dependency structures based on the correspondences it finds. The similarity of pairs of content words is measured by looking at a machine-readable Japanese–English dictionary and a thesaurus and it is the basis on which the algorithm operates. Matsumoto et al. note, however, that they do not plan to use their algorithm with complex sen-

tences, as it performed poorly in their initial testing. They suggest using their system for the matching between simple sentences or verb phrases only. Apart from that, it is unclear to what extent proper sentences can be generated from the dependency structures they use, and so the usability of the constructed parallel treebanks for data-oriented MT is questionable.

Meyers et al. (1998) work with what they call *regularised parses*, which are similar to the F-structures of LFG, but use dependency structures instead. The presented system is built on the foundation presented in (Grishman, 1994, Meyers et al., 1996). Meyers et al. use a greedy search-based algorithm to align these structures, which is very similar to the algorithm described in section 3.1.2, although the scoring algorithm they use is recursive and much more complex than the one from section 3.1.1 in that it incorporates features of the dependency structures in addition to word-level translation data obtained from bilingual dictionaries. Another significant difference is that they allow many-to-many alignments between trees. This is done in order to align all nodes in the trees, so that the aligned substructures can be extracted and used as transfer rules in an MT setting. In fact, the system does not output a parallel treebank, but rather proceeds directly with the extraction of transfer rules after the alignment, which is its main purpose. If it could be altered to produce a parallel treebank as output, it presents a promising solution to the problem of aligning dependency trees. It is, however, unclear whether the algorithm can be adapted to operate on phrase-structure trees.

Menezes and Richardson (2003) present a rule-based system for the alignment of *logical forms* (LFs) presented as dependency structures (similar to the ones used by Meyers et al. (1998)). For the word-alignment information they use an extensive electronic bilingual dictionary augmented with statistically acquired translational correspondences. This word-alignment data is used to provide initial hypothetical alignments for a tree pair. The

system then proceeds through a list of 18 rules that determine whether a hypothetical link is to be realised or not and that may add new links that have not been suggested by the word-alignment data. The authors claim that their system is language pair-independent, but this relies solely on the fact that LFs should contain roughly the same relation labels regardless of the language involved, rather than on some property of the system itself. This might prove to be problematic when switching to a new data-set and/or language pair. In addition, such a design decision makes the system unsuitable for the alignment of other types of structures. An additional drawback of this system is the fact that it allows one-to-many and many-to-one alignments, which—as discussed in section 3.1—do not necessarily add new information to the aligned tree pair.

Eisner (2003) proposes the use of Synchronous Tree Substitution Grammars (STSGs) for the training of MT systems on pairs of trees. He uses dependency structures in his paper, but the system can also be used with phrase-structure trees or even tree forests. The presented system operates by first hypothesising all possible derivations of the source and target trees in a pair, as well as all possible alignments between the resulting elementary trees. Versions of the inside-outside algorithm (Baker, 1979) and Expectation Maximisation algorithm (EM – Dempster et al., 1977) are used to calculate statistics on the occurrence of elementary tree-pairs. This setup, however, finds the joint probability of the occurrence of a source-target pair of trees, summed over all possible alignments between the trees. A *1*-best Viterbi (1967) modification of the algorithm can be used to produce the single best derivation of the tree-pair, that would correspond to the optimal alignment. Such a *1*-best Viterbi STSG system—if implemented to work with phrase-structure trees—will function quite similarly to the string-to-string alignment module presented in section 3.2, but, due to its use of the EM algorithm, it is computationally much more complex.

Ding et al. (2003) present a lexical alignment algorithm that uses dependency structures to supply the reordering information for the word alignments. Although the algorithm operates by introducing links between nodes in the dependency structures, these structures are deconstructed into sub-graphs (called *treelets* by the authors) in the process. Thus the output of this system consists of word and phrase alignments derived from linked *treelets*, rather than of a parallel treebank.

2.1.3. Phrase-Structure Trees

An early algorithm for the alignment of phrase-structure trees is presented by Kaji et al. (1992). The authors use the language pair English–Japanese, but the algorithm is designed to be language pair-independent. Before proceeding to the actual alignment, both sentences in a sentence pair are syntactically analysed using a chart parser. The eventual parse ambiguities that might occur during parsing are supposed to be preserved in the parse charts; they will be resolved during the alignment process. For the alignment itself, first a bilingual dictionary is used to find potential content word matches between the source and target sentence words (function words are ignored). Potential ambiguities are left intact and resolved later.

The final stage is the alignment of phrases. This is done heuristically based on existing word-level matches, i.e. a target phrase can be aligned to a source phrase if and only if it contains matches for all the words in the source phrase and no matches to words outside of the source phrase. This process can resolve word-level ambiguities by aligning phrases unambiguously based on other content words. Also, if there is an ambiguity between phrases in (say) the source parse chart, only that phrase will be aligned that has a counterpart in the target sentence, thus implicitly resolving the ambiguity.

The system of Kaji et al. (1992) bears many similarities to the one presented in this work. However, it is purposely built for the extraction of translation templates for use in EBMT systems and, thus, does not produce

parallel treebanks. It is not even clear whether the algorithm can resolve all parse ambiguities in a sentence pair. Furthermore, the algorithm depends greatly on the quality of the bilingual dictionary, so a high level of lexical ambiguity in the dictionary might leave the algorithm with insufficient information about the alignment of certain sentence pairs. This promising algorithm is one of the first of this type and later in this section I will discuss a few similar algorithms.

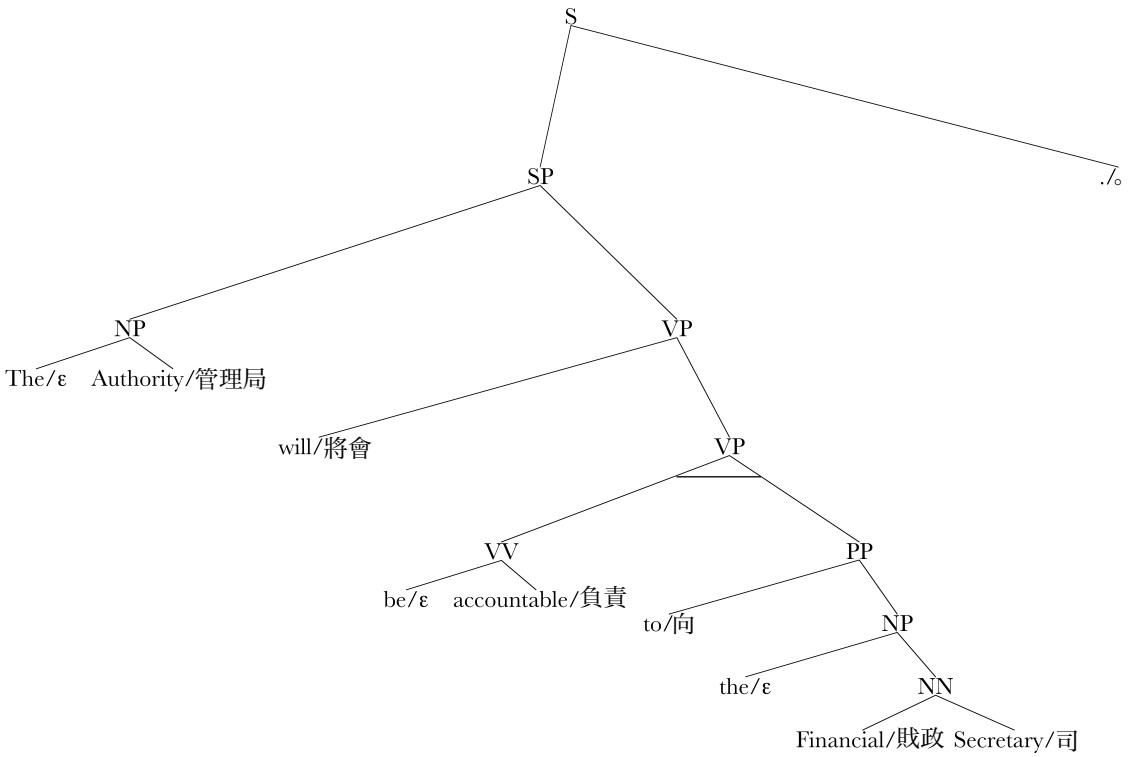


Figure 7: An English–Chinese ITG parse tree

Wu (2000) proposes to use Stochastic Inversion Transduction Grammars (SITGs – Wu, 1997) to perform phrasal alignment. This method is most closely related to the string-to-string algorithm presented in section 3.2. The ITG model is used to generate a common analysis tree for the source and target sentences of a sentence pair, in which nodes might be marked to allow for the inversion of the surface order of their children when transitioning from the source to the target language. The leaves of such an analysis tree contain source/target word pairs and either the source or target word can be ε, which signifies an insertion or a deletion depending on the

particular case. Thus, each node in the tree spans both a source and a target phrase, which implicitly encodes the phrasal alignments. In Figure 7 we reproduce Figure 2 from (Wu, 2000: 143). Here an inversion is annotated as a horizontal line at the lower VP node. This means that, while the English parse is read in the traditional depth-first left-to-right manner, at this node for the Chinese side the right tree is traversed before the left, putting the Chinese equivalent of the English phrase *be accountable* at the end of the Chinese sentence.

A significant problem with this approach from my point of view is the fact that it imposes isomorphism between the phrase structures of two languages, which—even while allowing for surface order switching—can be undesirable for languages that vary significantly in their grammatical structures. Wu himself discourages the use of ITG for free-word-order languages, because it is often impossible to impose the necessary level of isomorphism, even while allowing inversions. Another drawback is that this system cannot make use of existing monolingual parsers, but rather always has to generate the structural analysis itself.

Lü et al. (2001) present an algorithm similar to (Wu, 2000), which is based on (Wu, 1995, 1997). They discuss the problems arising from the unavailability of large high-quality ITG grammars; they have to resort to the use of generic Bracketing Inversion Transduction Grammars ("BTGs"): simplified ITGs with only one non-terminal and no syntactic grammar. Lü et al., however, observe an unacceptable rate of ungrammatical phrase-pairs extracted using this method. Accordingly, they investigate options for the incorporation of syntactic information in the model via the monolingual parsing of the source and/or target sides of the data. One such possibility they discuss is the use of an English parse as a guide to the SITG in an English–Chinese setting. However, they still allow the SITG to take precedence when the structural difference between the guiding English parse and the Chinese structure proposed by the SITG is too big.

Lü et al. (2001) also look at the introduction of parses for both sides of the corpus using both a full Chinese parser and a simplified, but more accurate, base-phrase parser for Chinese. They present experiments judging the grammaticality of the structures generated by their system against a manually annotated test set, with respect to what syntactic information is added to the SITG. The best results are achieved by using the English parser and the base-phrase Chinese parser. I believe that this is due to the fact that the outputs of these two parsers exhibit the highest level of isomorphism. Thus, I expect such a system to perform poorly for languages with structurally highly disparate grammars. Another issue I find with this system is that it does not preserve the original syntactic structures that are available for the two languages, but rather uses them only as a constraint for the generation of an independent structure.

A different approach to the alignment of phrase structure trees is presented in (Imamura, 2001). Imamura uses standard methods in order to obtain word-alignment information for the sentence pair being analysed. Then this information is used to heuristically locate translationally equivalent phrases. In order for two phrases to be translationally equivalent, they should have the same (or similar enough) syntactic categories and convey the same semantic information ("Words in the pair corresponded with no deficiency and no excess" (Imamura, 2001: 378)). The biggest issue with this approach lies in its reliance on syntactic categories. In fact, for the English–Japanese data that Imamura discusses, seven non-terminal categories had to be created that would be used to categorise all available non-terminal categories for the English and Japanese data. This means that for each new language pair these categories might need to be redefined. In addition, such an approach makes it impossible to account for phenomena such as nominalisation across languages, where a VP in one language is realised as an NP in another. An interesting idea discussed by Imamura (2001) is the use of the alignment algorithm for disambiguation during

parsing. The described procedure uses the number of aligned sub-structures as the measure that decides ambiguous cases. Imamura also proposes a method for the combination of partial parse trees resulting from an incomplete parse.

Gildea (2003) proposes a tree-based translation model that integrates sub-tree alignment in the translation process. It is based on a tree-to-string model that employs reordering and cloning operations on a source-language tree to come up with a tree structure that will have the correct target-language word order. The tree-to-tree model from Gildea (2003) differs from this tree-to-string model in that the transformations of the source tree should be guided by the existing target tree. It encodes the transformations needed to change a source-language tree to a target-language tree (similarly to an STSG), rather than information about alignment between nodes in the existing trees. Such information is only generated temporarily and used to decide on the proper transformations to be applied. Additionally, some adjustments are allowed to be made to the target tree to diversify and simplify the transformation steps needed to be performed to the source tree. Due to the tight integration of the alignment algorithm in the translational model, I do not see this system as a viable option for the generation of parallel treebanks.

Groves et al. (2004) present a rule-based algorithm inspired by (Menezes and Richardson, 2003), but adapted for use on phrase-structure trees. Rather than using a pre-developed bilingual lexicon for the word-alignment data, as Menezes and Richardson do, Groves et al. obtain a probabilistic bilingual dictionary using an automatic tool. They also exclude functional words from the word-alignment process to improve performance and decrease lexical ambiguity. Further, their algorithm closely follows the one by Menezes and Richardson (2003). There are a few additional differences, however. First, they do not allow one-to-many and many-to-one alignments,

as such alignments are incompatible with the DOT paradigm (Hearne and Way, 2006) and do not necessarily add information. Second, they only use five alignment rules versus the 18 used by Menezes and Richardson (2003). Still, the system of Groves et al. (2004) suffers from the same major drawback as that of Menezes and Richardson. Although it is also designed to be language pair-independent, many of its rules depend on the non-terminal labels in the trees to function. As for (Menezes and Richardson, 2003), this requires a readjustment of the system not only for each new language pair, but even also for a new syntactic analysis for a known language pair.

An interesting tree-to-string heuristic alignment algorithm operating on phrase-structure trees is described by Ambati and Lavie (2008). It has been designed for the direct induction of a translation model and it is unclear whether it can produce parallel treebanks in its current form. More relevant is their tree-to-tree model *TnT*. Even though it is derived from their tree-to-string model *TnS*, it closely resembles the system by Kaji et al. (1992). A major distinction is that it uses statistical tools to acquire the word alignments that seed the sub-tree alignment, rather than a bilingual dictionary. It is also very similar to the system of Meyers et al. (1998), with the major difference being that the algorithm of Meyers et al. operates on dependency structures. Ambati and Lavie present results following the reasoning by Koehn et al. (2003: 50) that suggest that their *TnT* algorithm performs much worse than their *TnS* algorithm in a Phrase-Based SMT task. This leads them to the development of a *TnT′* algorithm, that uses transformations similar to those discussed by Gildea (2003) to impose a level of isomorphism between the trees in a tree pair.

The *TnT* algorithm of Ambati and Lavie (2008) is an improvement on the system of Kaji et al. (1992), due to its use of automatically generated word-alignment data. However, it depends greatly on the ability of the word-alignment software used to produce the best set of alignments for a

particular sentence pair. This means that any error in word alignment will be propagated directly to the sub-tree level. My algorithm, on the other hand, uses statistics extracted from the word-alignment data so that an alignment error in a particular sentence pair could be superseded by the overall statistics and thus will not influence the sub-tree alignments significantly (cf. sections 3.1.1 and 4.2). The *TnT′* algorithm presented is not suitable for the generation of parallel treebanks, as it alters the target-language tree structures. The *TnS* algorithm, if it could be used to generate parallel treebanks, might be the most promising of the three and an alternative to the tree-to-string method I describe in section 3.2. However, it will also suffer from the direct use of word-alignment data, as the *TnT* algorithm would.

2.2. Conclusions

In this chapter I looked at existing parallel treebanks and at sub-tree alignment techniques that could be used for the generation of parallel treebanks.

I found that of the few parallel treebanks that could be used effectively in real-life MT applications (Čmejrek et al., 2004, Uchimoto et al., 2004, Samuelsson and Volk, 2006), none are of sufficient size. This is generally attributable to the fact that the manual annotation of parallel treebanks is a very time-consuming and laborious task, which is prone to errors and inconsistencies between annotators. Thus, I do not expect to see large-scale hand-crafted parallel treebanks soon.

A summary of the findings in section 2.1 is given in Table 1. Han et al. (2002) do not provide sufficient information about their treebank to fill in the table, while the work of Megyesi et al. (2008) is in too early stage to be properly judged. As the PCEDT (Čmejrek et al., 2004) does not provide sub-sentential alignments directly, but rather tools to build them, the requirement for no one-to-many links cannot be directly applied. Uchimoto

et al. (2004), Hansen-Schirra et al. (2006) and Cyrus (2006) do not use syntactic labels in the usual sense and Uchimoto et al. and Cyrus do not provide concrete information on the size of their parallel treebanks.

As will be discussed in section 4.1, my system fares favourably to the presented parallel treebanks with respect to the criteria stated in the beginning of this chapter. The system does not allow one-to-many structural links; it operates independently of the labelling schema used without referring to any language-specific features of the data; the generated parallel treebanks are generic enough so that they can be used with little difficulty for different tasks. Also, the system can annotate significantly more data than is present in any of the reviewed hand-crafted parallel treebanks within only a few days.

Features	(Samuelsson and Volk, 2006)	(Uibo et al., 2005)	(Han et al., 2002)	(Čmejrek et al., 2004)	(Marinov, —)	(Megyesi et al., 2008)	(Uchimoto et al., 2004)	(Ahrenberg, 2007)	(Hansen-Schirra et al., 2006)	(Cyrus, 2006)	my system
no one-to-many links	✗	✓	?	n/a	✗	?	✓	✓	✗	✗	✓
labelling-independent	✓	✗	?	✓	✓	?	n/a	✗	n/a	n/a	✓
no language-specific annotation	✓	✓	?	✓	✓	?	✗	✓	✓	✓	✓
task-independent	✓	✓	?	✗	✓	✗	✓	✗	✗	✗	✓
size (sentence-pairs)	1000	>500	>5000	>21,000	200	≈300,000 tokens	?	1200	≈1mil tokens	?	>1mil

Table 1: Realisation of expected features of parallel treebanks

From the sub-tree alignment systems I reviewed, I did not find any designed for the generation of parallel treebanks. Most of the systems were developed for the task of translation-template extraction, while some are used to guide and/or disambiguate a phrase or word-alignment algorithm. There were also a few systems used for disambiguation during parsing. Still, a number of the presented systems can (to a certain extent) be converted for

use in the generation of parallel treebanks. Most appealing and closely related to my system are the systems from (Kaji et al., 1992, Eisner, 2003, Ambati and Lavie, 2008). Of these, Eisner's system uses EM to find the best set of alignments for a given tree pair. Although promising, such an approach is much more computationally complex than the one I developed. The system of Ambati and Lavie is generally an improvement over that of Kaji et al., but both systems suffer from the same problem; they both rely on perfect word-alignment data on a per sentence-pair basis, which means that every sentence pair for which an error in word alignment occurs will have erroneous non-lexical alignments as well.

Features	(Sadler and Vendelmans, 1990)	(Matsumoto et al., 1993)	(Meyers et al., 1998)	(Menezes and Richardson, 2003)	(Eisner, 2003)	(Ding et al., 2003)	(Kaji et al., 1992)	(Wu, 2000)	(Lü et al., 2001)	(Imamura, 2001)	(Gildea, 2003)	(Groves et al., 2004)	(Ambati and Lavie, 2008)	my system
preserve original trees	✓	✗	✓	✓	✓	✗	≈	n/a	✗	≈	✗	✓	✓	✓
no one-to-many links	✓	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓
theory-independent	✓	✗	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓
labelling-independent	n/a	n/a	✗	✗	✓	✓	✓	✓	✓	✗	✓	✗	✓	✓
language-independent	✗	✓	✗	✗	✓	✓	✓	✓	✓	✓	✓	✗	✓	✓
robust wrt word alignments	✓	✗	✓	✓	✓	✗	✗	✓	✓	✓	✓	✓	✗	✓
task-independent	✓	✗	✗	✗	✓	✗	✗	✓	✓	✓	✗	✓	✗	✓
produces parallel treebanks	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✗	✓	≈	✓
computationally efficient	✗	✗	✗	✓	✗	✓	✓	✓	✓	✓	✓	✓	✓	✓

Table 2: Realisation of expected features of sub-tree-alignment systems

A summary of the presented in section 2.2 existing sub-tree alignment systems is given in Table 2. Sadler and Vendelmans (1990) and Matsumoto et al. (1993) do not align labelled data and, thus, cannot be judged with respect to labelling-schema independence. The systems presented by Kaji et al. (1992) and Imamura (2001) incorporate parse-tree disambiguation. While this technically constitutes a modification of the original structures, it cannot necessarily be viewed as a negative feature. The work of Wu (2000), on the other hand, entails the generation of aligned syntactic structures from plain data and, therefore, it cannot be classified according to its preservation of original structures. As a final remark on Table 2, although the system developed by Ambati and Lavie (2008) is not designed to produce parallel treebanks, they did supply a version of the HomeCentre corpus aligned using their methods (see section 4.1 for details), which I used in the evaluation that I present in Chapter 4.

Thus, I have been unable to find a robust and efficient tool for the generation of parallel treebanks. This, combined with the scarcity of hand-crafted parallel treebanks, makes it an important task to develop such a tool, which would enable the generation of high-quality training resources for use within the field of syntax-based MT.

In the next chapter, I present the design of the system for the automatic generation of parallel treebanks that I developed. This system addresses the drawbacks I found in the algorithms discussed in this chapter and—as will be seen in Chapter 4—produces high-quality parallel treebanks[3] that can lead to better performance in machine translation tasks compared to their hand-crafted counterparts.

[3] Due to the MT background of the project within which the presented work was undertaken, the term *high-quality parallel treebank* should be considered in the context of the use of a parallel treebank as a training resource for statistical MT systems.

3. System Design

In the previous chapter, I looked at existing hand-crafted parallel treebanks and at several different strategies for solving the task of sub-sentential alignment. I came to the conclusion that a new approach to the generation of parallel treebanks is needed, as the existing hand-crafted treebanks are of insufficient quality and none of the existing sub-tree alignment algorithms have been developed into systems that allow for the generation of parallel treebanks from parallel corpora.

In this chapter, I present the algorithms that lie at the foundation of my sub-tree alignment system, as well as the various features and configuration options of the system. The software presented here enables the automatic generation of parallel treebanks from parallel corpora using minimal external resources. The only other tool that is required is a word-alignment tool (eg. GIZA++ – Och and Ney, 2003). However, if monolingual phrase-structure parsers[4] or at least Part-Of-Speech (POS) taggers exist for any of the languages in question, they can be used to pre-process the data.

The basic functionality of the sub-tree alignment software described in this chapter was designed in the 'ATTEMPT' project at the National Centre for Language Technology jointly with John Tinsley, Mary Hearne and Andy Way (Tinsley et al., 2007), namely the scoring functions, the basic greedy-search-based selection algorithm and the *skip1* and *skip2* disambiguation modules. The software system itself, as well as the more advanced alignment modules (*span1* and *rescore* modules and string-to-string, string-to-tree and tree-to-string alignment) were developed by the author of this work and released as an open-source tool at the Open-source Convention at the Third MT Marathon in Prague, the Czech Republic (Zhechev, 2009).

[4] Henceforth, I will use 'parser' to mean 'monolingual phrase-structure parser', unless stated otherwise.

The full-search-based selection algorithm was designed in co-operation with Khalil Sima'an of the University of Amsterdam.

In all use cases for the software system, a word-alignment tool is used first to obtain word-alignment probabilities for the parallel corpus in question for both language directions. I will start with the description of the case in which parsers are available for both languages, as this is the core of the system: a tree-to-tree aligner (Tinsley et al., 2007). The parsers are used to parse both sides of the parallel corpus in a pre-processing step. The resulting parsed data together with the word-alignment probability tables are then used as the input to a sub-tree alignment system that introduces links between nodes in corresponding trees according to their translational-equivalence scores. The output of the sub-tree aligner is the desired parallel treebank.

If there is no parser available for one of the languages, the parallel corpus—together with the word-alignment tables—is fed directly to a modified version of the sub-tree aligner that can produce unambiguous parallel treebanks from plain data.

I will now look at the alignment algorithms in greater detail, starting with the tree-to-tree alignment and then moving on to the string-to-string, string-to-tree and tree-to-string cases.

3.1. Tree-to-Tree Alignment

First, following the criteria I set up in Chapter 2, the tree-to-tree aligner has to follow certain principles to fit into the above framework:

- Independence with respect to language pair, constituent-labelling scheme and POS tag set. Any language-dependence would require human input to adjust the aligner to a new language pair.
- Preservation of the original tree structures. I regard these structures as accurate encodings of the languages, and any change to them might distort the encoded information.

- Dependence on a minimal number of external resources, so that the aligner can be used even for languages with few available resources.
- The word-level alignments should be guided by links higher up the trees, where more context information is available.

These principles guarantee the usability of the algorithm for any language pair in many different contexts. Additionally, there are a few well-formedness criteria that have to be followed to enforce feasible alignments, including:

- A node in a tree may only be linked once.
- Ancestors of a source linked node may only be linked to ancestors of its target linked counterpart. By analogy, this also holds for descendants.

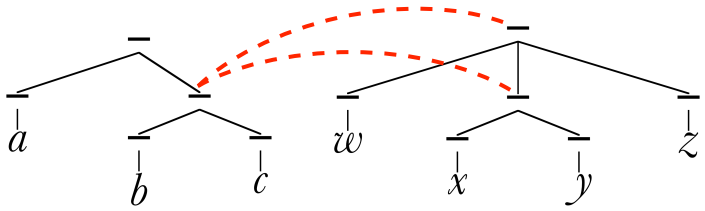


Figure 8: An example of alignment hypotheses incompatible due to linking the same node in a tree

In Figure 8, you see a pair of links that contravene the first well-formedness criterion above. The two links presented in red cannot coexist, as they both link the same node in the source tree. If they were to be valid at the same time, the string ⟨*b c*⟩ would have to be translationally equivalent both to the string ⟨*x y*⟩ and to the string ⟨*w x y z*⟩ within the context of the same sentence pair. This is an undesirable situation and is, therefore, prohibited in my system.

Figure 9 exemplifies a contradiction to the criteria governing the ancestry of aligned nodes outlined above. In this case, while the node s_1 is an ancestor of the node s_2, the nodes t_1 and t_2 are siblings. Establishing both red links at the same time would result in the string ⟨*b*⟩ translating as ⟨*w*⟩ and the string ⟨*b c*⟩ translating as ⟨*x y*⟩. However, the string ⟨*w*⟩ is not a part

of the string $\langle x\ y \rangle$, while the string $\langle b \rangle$ is a part of its translation $\langle b\ c \rangle$. Therefore, such translational equivalencies cannot coexist within the same context.

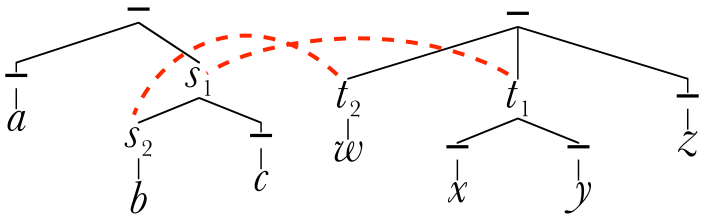


Figure 9: An example of alignment hypotheses incompatible due to a conflict in the ancestry relations between nodes

Similar criteria appear in many papers on sub-tree alignment (Sadler and Vendelmans, 1990, Kaji et al., 1992, Matsumoto et al., 1993, Grishman, 1994, Wu, 1997), although they are rarely stated explicitly. A notable exception is Wu (1997), with his definition of *crossing constraints*. Albeit very similar to the criteria stated above, Wu's *crossing constraints* are used to restrict bilingual parsing, rather than to guide sub-sentential alignment.

Links produced according to the well-formedness criteria listed above encode enough information to allow the inference of complex translational patterns from a parallel treebank, including some idiosyncratic translational divergences, as discussed by Hearne et al. (2007). In what follows, a hypothesised alignment is regarded as incompatible with the existing alignments if it violates any of these criteria.

The sub-tree aligner operates on a per sentence-pair basis and each sentence-pair is processed in two stages. First, for each possible hypothetical link between two nodes, a translational-equivalence score is calculated. Only the links for which a nonzero score is calculated are stored for further processing. Unary productions from the original trees, if available, are collapsed to single nodes, preserving all labels (see Figure 10). Thus, the aligner will consider a single node—instead of several nodes—for the same lexical span. This does not reduce the power of the aligner, as the translational-equivalence scores are based on the surface strings and not on the tree structures.

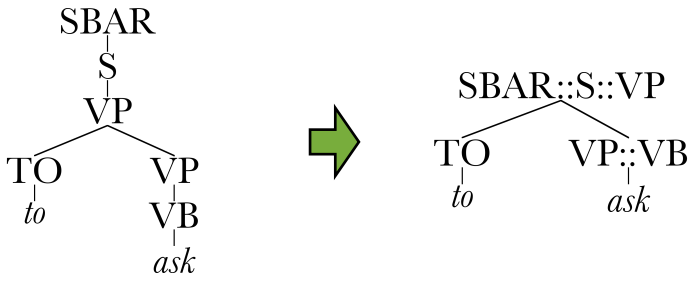


Figure 10: Example of collapsing unary nodes in a tree structure

During the second stage, the optimal combination of links is selected among the available nonzero links. The selection can be performed using either a greedy search, or a full search for the best combination.

I will first discuss the formulae according to which the translational equivalence scores are being calculated and then I will turn to the presentation of the two selection algorithms.

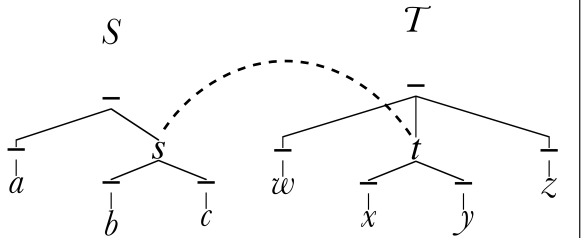
3.1.1. Translational Equivalence

Given a tree pair $\langle S, T \rangle$ and a hypothesis $\langle s, t \rangle$, first the strings in (1) are computed, where $\langle s_i \ldots s_{ix} \rangle$ and $\langle t_j \ldots t_{jy} \rangle$ denote the terminal sequences dominated by the nodes s and t respectively, and $\langle S_1 \ldots S_m \rangle$ and $\langle T_1 \ldots T_n \rangle$ denote the terminal sequences dominated by the root nodes of the trees S and T. Here, *inside* are the strings that represent the spans of the nodes being linked and *outside* are the strings that lay outside the spans of those nodes. These string computations are illustrated in (2), where the inside strings for the represented link are $\langle b\ c \rangle$ and $\langle x\ y \rangle$ for the source and target sides respectively; the outside strings are $\langle a \rangle$ and $\langle w\ z \rangle$ respectively.

(1)

	inside	*outside*
s_l	$\langle s_i \ldots s_{ix} \rangle$	$\bar{s}_l = \langle S_1 \ldots s_{i-1} s_{ix+1} \ldots S_m \rangle$
t_l	$\langle t_j \ldots t_{jy} \rangle$	$\bar{t}_l = \langle T_1 \ldots t_{j-1} t_{jy+1} \ldots T_n \rangle$

(2)



$s_l = \langle b\ c \rangle$
$t_l = \langle x\ y \rangle$
$\bar{s}_l = \langle a \rangle$
$\bar{t}_l = \langle w\ z \rangle$

The score for the given hypothesis is $\gamma(\langle s, t\rangle)$ and is computed according to the formula in (3), that is by multiplying the word-level alignment probabilities for the inside and outside strings in both language directions. For the calculation of the word-level translational-equivalence scores $\alpha(x|y)$ in (3) either *score1* (4) or *score2* (5) can be chosen. Here, x and y represent corresponding strings on the source and target side. Although they are variables, to simplify the exposition I will consider x to be the source-side string and y to be the target-side string.

The scoring formula *score1* was the first one that was developed by our team as an easy and intuitive way to calculate string-based translational equivalence. However, we realised that this formula had certain fundamental drawbacks, as for example a strong bias towards short strings. Therefore, we developed *score2*, based on IBM Model 1, which decreases the bias from *score1* significantly. In the present work I am including both formulae, as both of them performed satisfactorily in initial testing and further evaluation was needed in order to decide which will lead to better results. Of course, the modular nature of the system allows the implementation of new scoring formulae, should this be desired.

(3) $$\gamma(\langle s, t\rangle) = \alpha(s_l|t_l)\cdot\alpha(t_l|s_l)\cdot\alpha(\bar{s}_l|\bar{t}_l)\cdot\alpha(\bar{t}_l|\bar{s}_l)$$

To calculate $\alpha(y|x)$ for a target string y given a source string x according to (4), for each source token x_i you first sum the word-alignment probabilities of the target tokens y_j, given the source token. This gives you the probability masses of the target string corresponding to each of the source tokens and multiplying these gives you the alignment probability. In (5), the word-alignment probabilities are used to obtain an average vote by the source tokens for each target token y_j. Then the product of the votes for the target words gives the alignment probability $\alpha(y|x)$ for the two strings.

$$(4) \quad score1 \quad \alpha(y|x) = \prod_{i}^{|x|} \sum_{j}^{|y|} P(y_j|x_i)$$

$$(5) \quad score2 \quad \alpha(y|x) = \prod_{j}^{|y|} \frac{\sum_{i}^{|x|} P(y_j|x_i)}{|x|}$$

It should be noted, that for robustness, in the actual implementation of the scoring functions, I always add the *NULL* token to both the source and target strings. This allows the algebraic description of insertions and deletions. Such a setup, however, requires word-alignment probability tables that include probabilities for aligning words to the *NULL* token. The GIZA++ tables that I use for my experiments in Chapter 4 include such probabilities, but not for all tokens. Therefore, I had to devise a mechanism for dealing with the cases where the word-alignment probability of a certain token to *NULL* is unknown.

There are two possible cases here. A token that does not have an established word-alignment probability to *NULL* may or may not have a word-alignment probability to at least one of the other tokens in the target string. If it cannot be aligned to any of the target tokens, then its word-alignment probability to *NULL* is set at *1.0*, which adequately represents the fact that this token has to be an insertion, if the two strings are to be translationally equivalent. On the other hand, if the token in question can be aligned to at least one of the non-*NULL* tokens in the target string, then I deduce that it cannot be an insertion and I set to *0.0* its alignment probability to *NULL*. The word-alignment probability of aligning *NULL* to *NULL* is always set to *1.0* so that it does not affect the general translational-equivalence score computation.

The user has the option to select either *score1* or *score2* when using the sub-tree aligner. In Chapter 4, I will use the evaluation results to decide which of these two scoring formulae produces higher-quality alignments.

3.1.2. Greedy-Search Algorithm

The greedy-search algorithm is simple. The set of nonzero-scoring links is processed iteratively by linking the highest-scoring hypothesis at each iteration and discarding all hypotheses that are incompatible with it until the set is empty, cf. Figure 11.

```
while unprocessed hypotheses remain do
      link the highest-scoring hypothesis
      discard all incompatible hypotheses
end while
```

Figure 11: Basic greedy-search-based selection algorithm

An example which presents the process of alignment is shown in Figure 12. The numbers attached to the node labels are their IDs, used to identify the alignments. Along the left side of the table are the IDs for the source (English) tree, with the IDs for the target (French) tree along the top. All red squares represent alignment hypotheses that have received zero as a translation equivalence score according to the scoring mechanism detailed in section 3.1.1 and are, therefore, irrelevant for the alignment process. The green links between the trees are the alignment hypotheses that have been established by the selection process and correspond to the green cells in the alignment table, while the red ones are the hypotheses that have been discarded and correspond to the pink cells in the table. The numbers in the cells correspond to the serial number of the iteration of the selection process during which they were processed.

For example, the link represented by the green cell with the number one in Figure 12 (N-10 ⇔ A-8) was established during the first iteration of selection; at the same time the link represented by the pink cell with the number one (NP-8 ⇔ A-8) was discarded, because it includes the same target-tree node as the one that had just been linked, namely A-8. An explanation of the possible incompatibilities that may occur between possible links is given at the beginning of section 3.1. Further in the selection process, the hypothesis V-4 ⇔ V-3 is being linked without conflicts. The third

alignment that is established by the algorithm is ROOT-1 ⇔ ROOT-1 and there are two other hypotheses that are incompatible with it, both because they involve a node from the link that is being established. In this manner, the system selects six out of the ten available nonzero links, presenting them as the best set of alignments for this particular sentence pair.

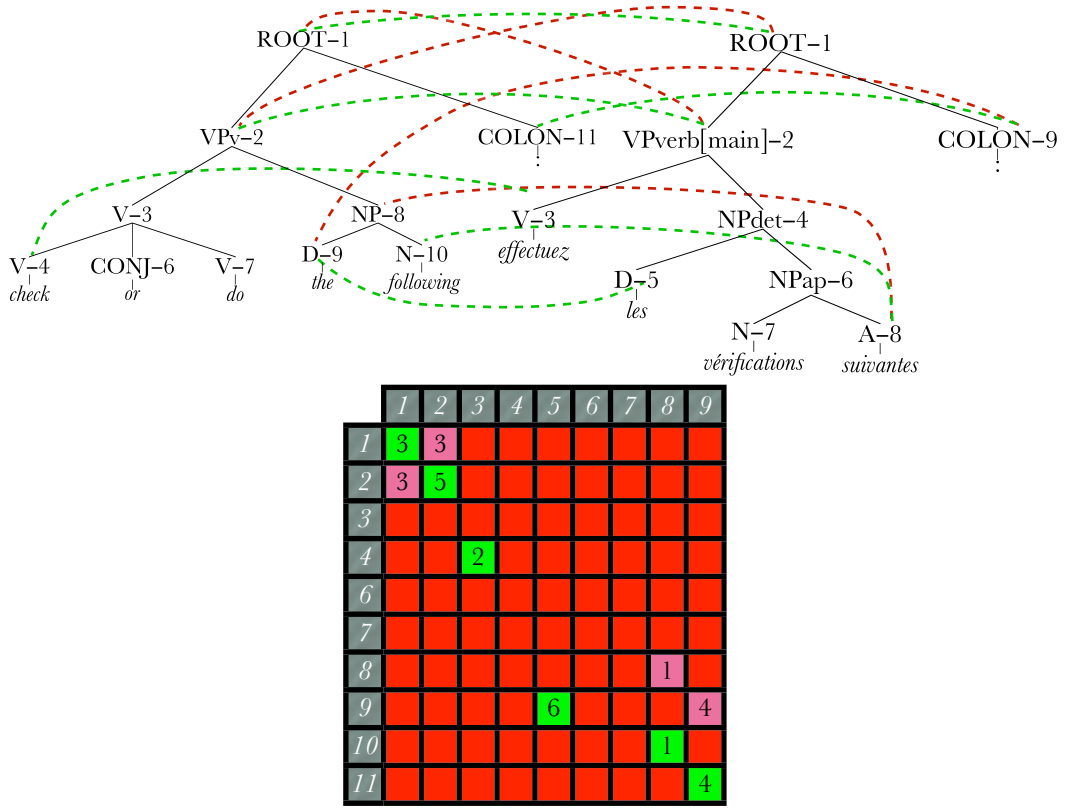


Figure 12: An aligned sentence pair and a table presenting the alignment steps taken by the greedy-search-based selection algorithm

Problems arise when there happen to be several hypotheses that share the same highest score. There are two distinct cases that can be observed here; these top-scoring hypotheses may or may not represent incompatible links, as can be seen in Figure 13.

If all such hypotheses are compatible like in the trivial case in the example, they are all linked at the same time and all remaining unprocessed hypotheses that are incompatible with any of those links are discarded. In Figure 13, you would link both **h** ⇔ **i** and **g** ⇔ **b**, and you would discard the link **a** ⇔ **b** as it shares the same target node as **g** ⇔ **b**. In case even one

among the top-scoring hypotheses is incompatible with the others like the non-trivial case in Figure 13, these hypotheses are skipped and processed at a later stage; the linking of some other hypotheses might result in the discarding of all skipped hypotheses but one, which will then be linked unambiguously. Here, if you could link **g ⇔ b**, you would have to discard **a ⇔ b** which shares the same target node and this would allow you to link **a ⇔ c**.

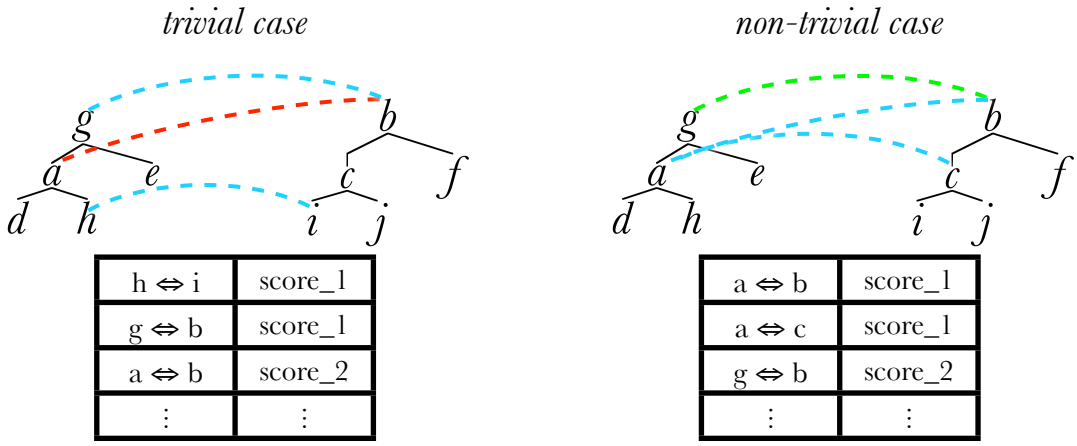


Figure 13: Two possible situations involving ambiguous links

The sub-tree aligner can be built to use one of two possible skipping strategies, which I call *skip1* and *skip2*. According to the *skip1* strategy shown in Figure 14, hypotheses are simply skipped until a score is reached, for which only one hypothesis exists (see Figure 15). This hypothesis is then linked and the selection algorithm continues as usual.

```
while unprocessed hypotheses with no tied competitors remain do
      while the highest-scoring hypothesis has tied competitors do
            skip all tied competitors
      end while
      link the highest-scoring hypothesis
      discard all incompatible hypotheses
end while
```

Figure 14: *skip1* selection algorithm

Figure 15 presents the steps of the *skip1* strategy for a group of hypothetical alignments. In this example, the first two hypotheses—**a ⇔ b** and **a ⇔ c**—share the same highest translational-equivalence score **score_1**, as seen in table *a*. As you see, however, both hypotheses involve linking node **a**

on the source side and, thus, cannot be linked at the same time. Therefore, in table *b* the first two rows are greyed out, as you skip them for the moment and look for the next highest score, which is **score_2**. There are no conflicts for the hypothesis **d ⇔ c** with the same score, so you link these nodes right away, as represented by the green colour in table *c*. The hypothesis **a ⇔ c** conflicts with **d ⇔ c**, however, and has a different score, so you need to block it at this step, which is why it is coloured red in table *c*. After you discard the two hypotheses that you made decisions on, you are left with the four hypotheses in table *d* and you can continue the selection process as before. In this example, this means that you can now link **a ⇔ b**, as there are no more conflicting hypotheses that share **score_1**, cf. table *e*.

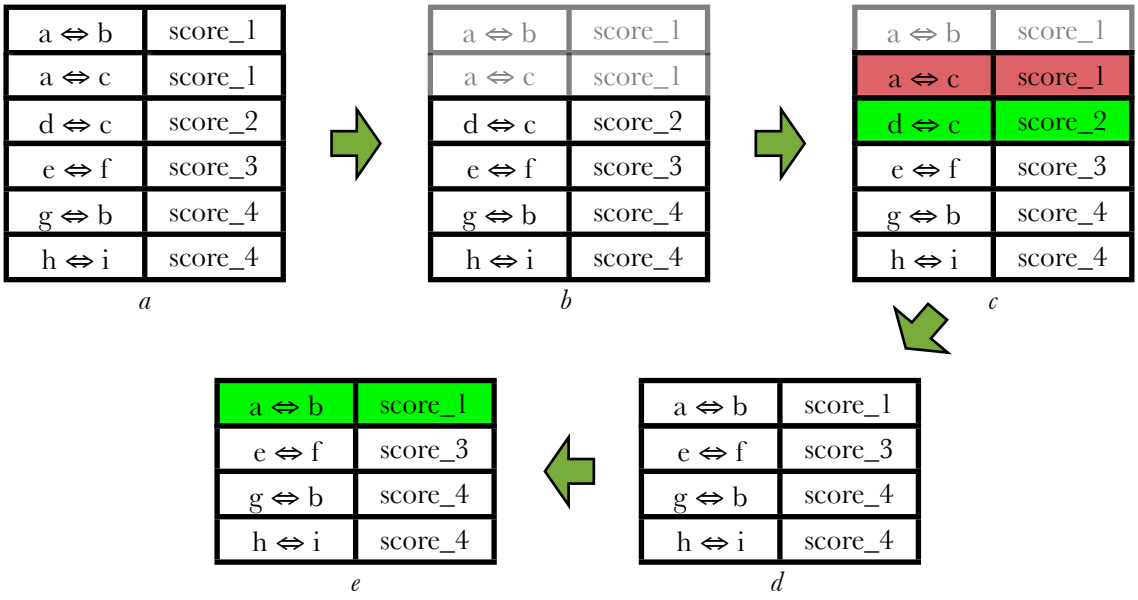


Figure 15: Step-by-step operation of the *skip1* algorithm

The *skip2* strategy is more complex, in that it also keeps track of which nodes take part in the skipped hypotheses, cf. Figure 16. When a candidate for linking is found, it is only linked if it does not include any of the involved nodes. The motivation behind this strategy is that a situation may occur in which a low-scoring hypothesis for a given constituent is selected in the same iteration as higher-scoring hypotheses for the same constituent were skipped, thereby preventing one of the competing higher-scoring hypotheses from being selected and resulting in an undesired link.

```
while unprocessed hypotheses with no tied competitors remain do
      if the highest-scoring hypothesis has tied competitors then
            mark the constituents of all tied competitors
      end if
      while the highest-scoring hypothesis has a marked constituent do
            skip
      end while
      link the highest-scoring hypothesis
      discard all incompatible hypotheses
      unmark all constituents
end while
```

Figure 16: *skip2* selection algorithm

An example of the operation of the *skip2* selection strategy can be seen in Figure 17. As in Figure 15, there is a conflict between the first two hypotheses, as they receive the same score. In this case, however, you first look at the nodes that are involved in these two hypotheses, namely **a**, **b** and **c**, and mark them in all hypotheses. In table *b*, the marked nodes are the red ones and you see that the following hypotheses have marked nodes: **a** ⇔ **b**, **a** ⇔ **c**, **d** ⇔ **c** and **g** ⇔ **b**. Next, you skip all hypotheses with marked nodes and the first hypothesis that is available for linking is **e** ⇔ **f**, as seen in table *c* where the first three rows are greyed out.

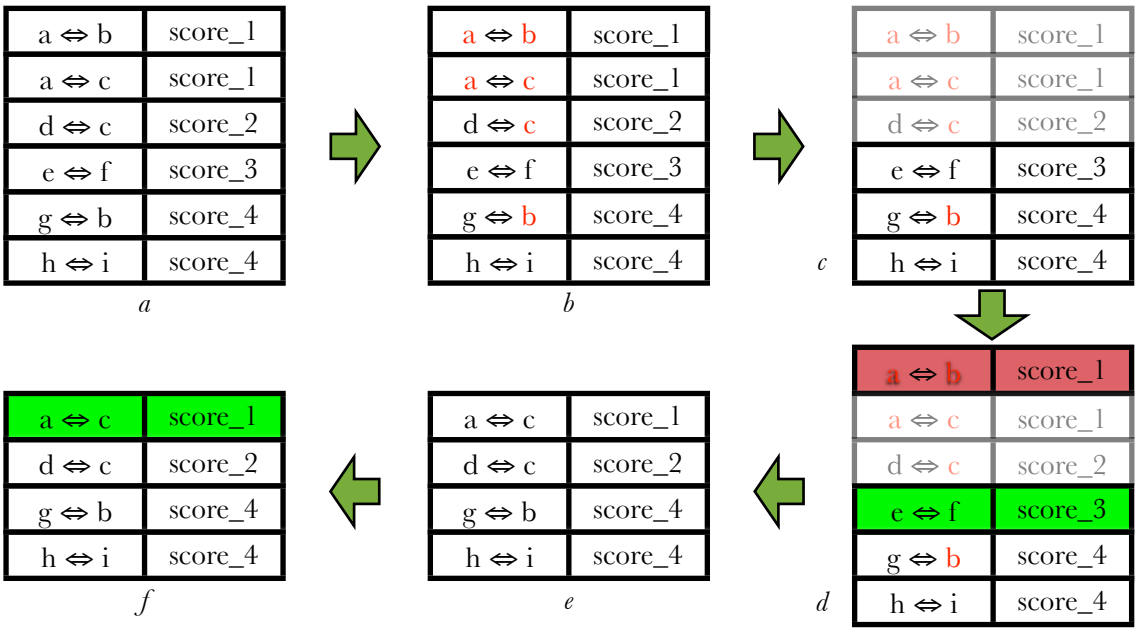


Figure 17: Step-by-step operation of the *skip2* algorithm

To be able to follow the selection process further, you need to look at the tree structures that are being aligned, which are presented in Figure 18.

There you notice that the hypothesis **e** ⇔ **f** is incompatible with the link between **a** ⇔ **b** (the red alignments in Figure 18), as the node **b** is an ancestor of node **f**. As discussed in the beginning of section 3.1, such two links cannot coexist and after you link **e** ⇔ **f**, you block **a** ⇔ **b**. You can see this situation in table *d* in Figure 17, where the successful link is coloured green and the blocked one is coloured red. In this way the hypothesis **a** ⇔ **b** is excluded from the further selection process and you are left with the four hypotheses in table *e*. Thus, the initial conflict is resolved and you can continue the selection process by linking the hypothesis **a** ⇔ **c** in table *f*.

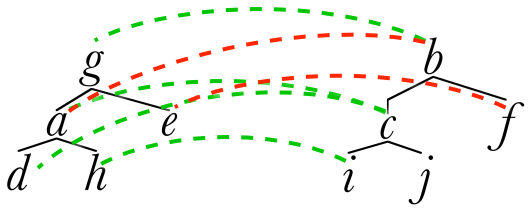


Figure 18: A graphical representation of the alignments from Figure 17

Regardless of whether *skip1* or *skip2* is used, sometimes a situation occurs in which the only hypotheses remaining unprocessed are equally likely candidates for linking according to the selection strategy. In such ambiguous cases our decision is not to link anything, rather than make a decision that might be wrong.

Initial testing of the aligner showed a peculiar trait; often lexical links would receive higher scores than the non-lexical links,[5] which sometimes resulted in poor lexical links preventing the selection of proper non-lexical ones. This went against my expectations that the non-lexical links would guide the selection of the lexical ones, as more context is available higher in the tree structures. The reason seemed to lie in short strings obtaining much higher translational-equivalence scores than average length strings, especially when using *score1*. Thus, the lexical links and a few top-level links tend to be produced first.

[2] *lexical* are such links, for which at least one of the linked nodes spans over only one word, see Figure 19.

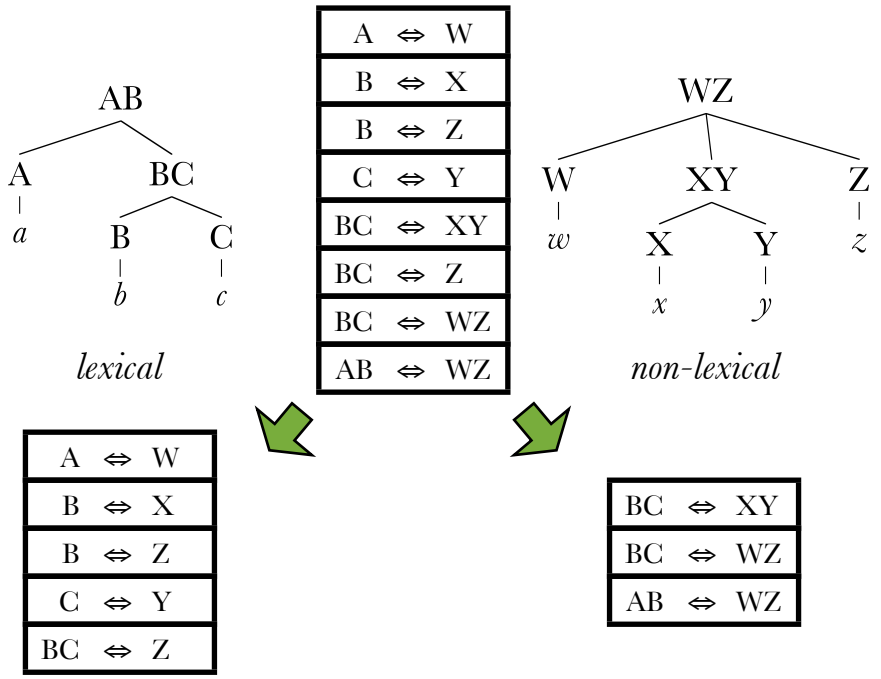


Figure 19: An example of *lexical* and *non-lexical* links for a tree-pair

This motivated the development of an extension to the selection algorithm, which I call *span1*. When enabled, this extension results in the set of nonzero hypotheses being split in two parts: one containing all hypotheses for lexical links, and one containing the hypotheses for non-lexical links, as shown in Figure 19. Consequently, links are first selected from the set of non-lexical links, and only when it is exhausted does the selection continue with the set of lexical links. This division does not affect the discarding of incompatible links after linking; incompatible links are discarded in whichever set they are found.

I will investigate the effectiveness of the *span1* module during the evaluation of the system in Chapter 4.

3.1.3. Full-Search Algorithm

The full-search algorithm was developed mainly to answer the question as to whether the greedy-search algorithm gives reasonable results and not some local maximum, as well as providing insights into whether we should not be doing a full search instead.

This is a backtracking recursive algorithm that enumerates all possible combinations of compatible links. For an explanation of the criteria

used to decide conflicts between links, refer to the beginning of section 3.1. A maximal combination of non-crossing links is such a combination of links for which any newly added link would be incompatible with at least one of the links already in the combination. All such maximal combinations found during the full search are stored for further processing.

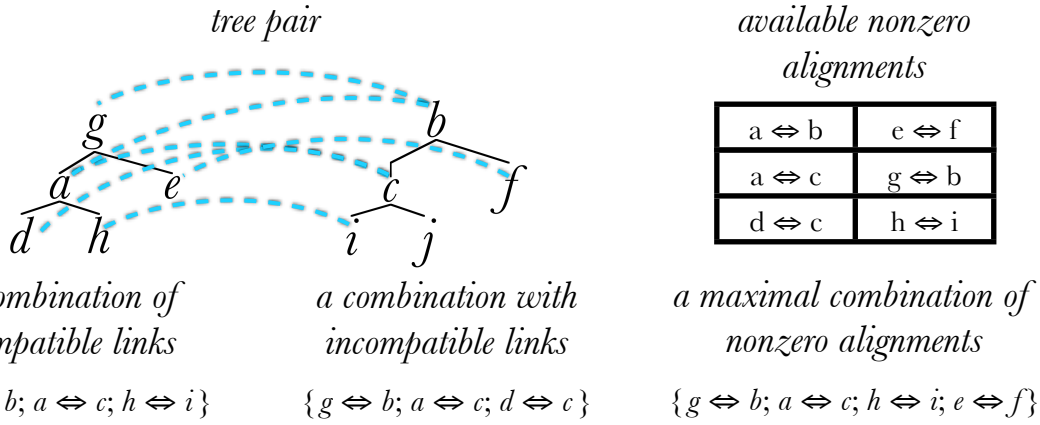


Figure 20: Possible combinations of sub-tree alignments for given tree pair and nonzero alignments

An example of possible link combinations is shown in Figure 20 using the tree pair and alignments from Figures 17 and 18. Here, the combination $\{g \Leftrightarrow b;\ a \Leftrightarrow c;\ d \Leftrightarrow c\}$ contains the incompatible links **a** ⇔ **c** and **d** ⇔ **c** which share the same target node and, therefore, this combination is not being considered by the full-search algorithm. The combination $\{g \Leftrightarrow b;\ a \Leftrightarrow c;\ h \Leftrightarrow i\}$, on the other hand, has no such conflicts and is among the ones enumerated by the algorithm. Still, this combination is not a maximal one, as the link **e** ⇔ **f** is compatible with all other links in the combination and can be added to it, resulting in the combination $\{g \Leftrightarrow b;\ a \Leftrightarrow c;\ h \Leftrightarrow i;\ e \Leftrightarrow f\}$. Out of the two remaining links, **a** ⇔ **b** is incompatible with **g** ⇔ **b** and the link **d** ⇔ **c** is incompatible with **a** ⇔ **c**. Therefore, the combination $\{g \Leftrightarrow b;\ a \Leftrightarrow c;\ h \Leftrightarrow i;\ e \Leftrightarrow f\}$ is one possible maximal combination of compatible links for the given tree pair. The remaining such combinations are $\{g \Leftrightarrow b;\ d \Leftrightarrow c;\ e \Leftrightarrow f\}$, $\{a \Leftrightarrow b;\ d \Leftrightarrow c\}$ and $\{a \Leftrightarrow b;\ h \Leftrightarrow i\}$.

After the full search is complete, the probability mass of each maximal combination is calculated by summing the translational-equivalence scores for all the links in the combination, where the scores are calculated using either the *score1* or the *score2* scoring strategy, discussed in section 3.1.1. The maximal combination of non-crossing links that has the highest probability mass is selected as the best alignment for the sentence pair.

e ⇔ f	.005
g ⇔ b	.004
d ⇔ c	.003
a ⇔ c	.002
a ⇔ b	.001
h ⇔ i	.001

Table 3: Hypothetical scores for the alignment hypotheses from Figure 20

Often, there are several distinct maximal combinations that share the highest probability mass; for longer sentences this number can rise to several hundred. One possible disambiguation strategy is to take the largest common subset of all maximal combinations. To exemplify this, I return to the example from Figure 20 and suppose that I have the hypothetical scores for the alignment hypotheses shown in Table 3. With such scores you have two maximal combinations of links with the same highest score of *.012*, namely $\{g \Leftrightarrow b; a \Leftrightarrow c; h \Leftrightarrow i; e \Leftrightarrow f\}$ and $\{g \Leftrightarrow b; d \Leftrightarrow c; e \Leftrightarrow f\}$. Here, the ambiguity is whether to establish the links **a** ⇔ **c** and **h** ⇔ **i**, or only the link **d** ⇔ **c** instead. The strategy employed by my system will only present the largest common subset of these maximal combinations as the proper set of alignments for the tree pair in question, namely $\{g \Leftrightarrow b; e \Leftrightarrow f\}$, thus discarding the ambiguous links.

Another possible strategy is to output all possible combinations and mark them as relating to the same sentence pair, thus leaving disambiguation to the task that uses the resulting aligned data. However, such a strategy depends on the requirements of the task using the resulting parallel

treebank and, therefore, cannot be properly developed without prior knowledge of these requirements. Still, the open-source nature of my system enables the development of the proper disambiguation strategy by the user, should it be required.

It should be noted, though, that this algorithm cannot be regarded as a feasible solution to real-life alignment tasks. In its nature, it is related to the Travelling Salesman Problem[6] (TSP – Whitney, 1932), albeit in a strongly restricted setting, which means that it has combinatorial complexity (cf. section 3.4). Still, on a small scale it can be used to evaluate the performance of the greedy-search algorithm, as will be discussed in section 4.1.3.

3.2. Other Alignment Modules

In this section I look at the string-to-string, tree-to-string and string-to-tree modules that are used when a parser is not available for one or both of the languages being aligned.

The string-to-string aligner can accept as its input plain or POS-tagged data. For a pair of sentences, all possible binary trees are first constructed for each sentence. All nodes in these trees have the same label (*X*) and are used as available link targets. In the case of POS-tagged data, the pre-terminal nodes receive the POS tags as labels.

After all link-hypothesis scores have been calculated according to the formulae in section 3.1.1, the string-to-string aligner continues with the selection of links in the same manner as the sub-tree aligner, with one extension; after a link has been selected—besides all incompatible links—all binary trees that do not include the linked nodes are discarded with any nonzero hypotheses attached to them. In this way, only those binary trees that are compatible with the selected links remain after the linking process. Figure 21 exemplifies this new type of incompatibility. The link between the

[6] The term 'Travelling Salesman Problem' is attributed to Hassler Whitney, although the problem was first formally described by Menger (1931) and has been discussed as early as the 1800s.

nodes X_2 and X_8 is incompatible with the link between X_3 and X_9, because no binary tree exists for the source string, of which both X_2 and X_3 can be part. On the other hand, the link between X_4 and X_8 is compatible with the link between X_3 and X_9, because X_4 and X_3 can be the children of X_1 in a binary tree covering the source string.

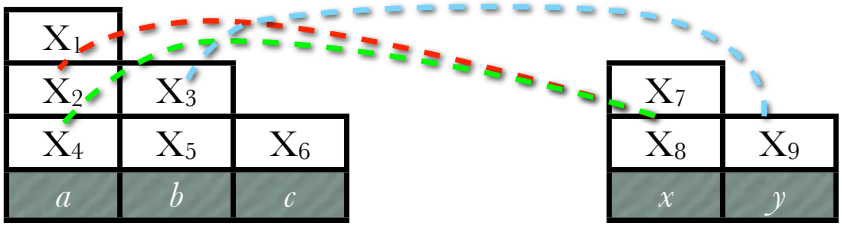


Figure 21: Link incompatibility during string-to-string alignment

In an additional step for the string-to-string aligner, all non-linked nodes (except for the root nodes) are discarded, thus allowing for the construction of unambiguous *n*-ary trees for the source and target sentences. If necessary, non-linked nodes are left intact to provide supporting structure in the trees. It is also possible to output a parse forest of all binary trees that are compatible with the alignments.

In its operation, the string-to-string aligner is very similar to Inversion-Transduction Grammars (ITG – Wu, 2000); however its goal is the generation of a parallel treebank as a training resource for machine translation, rather than the induction of a bilingual grammar for synchronous parsing.

The tree-to-string and string-to-tree modules differ from the string-to-string module in that a parser is available for one of the languages being aligned. In this case, the available parses are used, where available, instead of generating hypothetical binary trees. Also, at the output stage, the existing parses are preserved, except for any unary productions that are being collapsed as in the tree-to-tree alignment module. The non-parsed side may be POS-tagged, if a POS tagger is available. These two modules allow for the utilisation of all available resources for the generation of high-quality parallel treebanks.

3.3. Re-Scoring

It can be argued that each newly induced link in a sentence pair should affect the decisions regarding which links to select further in the alignment process for this sentence pair. This can be simulated to a certain extent using the simple re-scoring module discussed in this section.

The operation of this module relies on the fact that after a link has been introduced for a pair of trees, some of the word alignments available in the word-alignment tables for the tree pair will be incompatible with this link. Namely, these are the alignments between words within the span of the source node being linked and words without the span of the target node; as well as the alignments between words without the source node span and words within the target node span. For example, in Figure 22 the word alignments represented in green are compatible with the shown link, while the ones in red are incompatible and will be dropped by the re-scoring algorithm after the establishment of the link. The incompatibility arises because with the establishment of a link between the two trees, a certain context is established based on the fact that—in the case of Figure 22—the string $\langle b\ c \rangle$ translates as the string $\langle x\ y \rangle$. In such a context, an element outside $\langle b\ c \rangle$, namely $\langle a \rangle$, cannot be the translational equivalent of an element of $\langle x\ y \rangle$, and vice versa. The goal of this process is to make use of the implicit contextual information available in the phrase-structure trees.

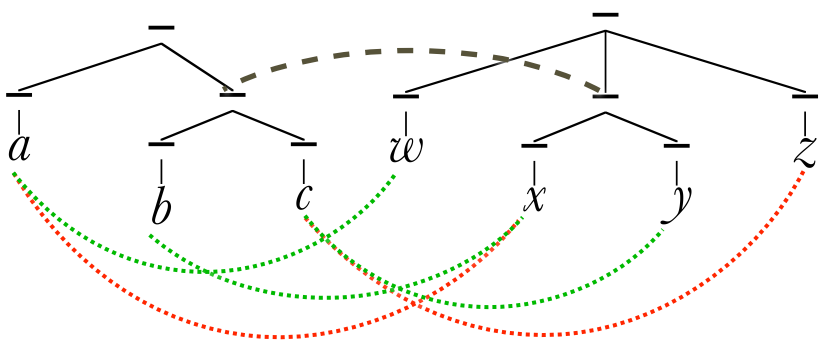


Figure 22: Compatible and incompatible word alignments, with respect to a particular sub-tree alignment

In this way, each time a new link has been selected, the incompatible word alignments are removed from the list of available word alignments for the tree pair and the scores of the remaining link hypotheses are recalculated. The linking process then continues as usual.

It should be noted that the re-scoring mechanism presented in this section can only be used in conjunction with the greedy-search-based selection. This is due to the fact that there is no inherent order for the links generated by the full-search algorithm and there is one for the greedy-based search.

3.4. Algorithm Complexity

In this section I will look at the complexity of the developed algorithms. There are two sides to this calculation that are equally important in our case: space complexity and time complexity.

3.4.1. Space Complexity

When looking at the space complexity, I am trying to judge the amount of memory that will be used when running the software. In my case, I need to store information about the tree structures that are being aligned and information about the links that are being generated.

Let us first look at the basic tree-to-tree case of alignment. Of the two space complexity-related variables, the second one depends on and is restricted by the first. Namely, in the final output, there can be at most as many links as there are nodes in the smaller of the two trees in a tree-pair. However, before selection begins, there may theoretically be many more hypotheses to store. Nevertheless, real-life experiments have shown that the number of nonzero alignment hypotheses (which are the only ones that I have to store in memory) tends to be very close to the number of nodes in the smaller tree. Compare for example Table 6 from Appendix A and Table 10 for the HomeCentre, where the average number of nodes for English trees is 15.33, while the average number of nonzero links induced using

score1 is 28.10; here the actual maximum number of links—given that the French trees have 17.52 nodes on average—will be 15.33×17.53=268.74, which is an order of magnitude larger than the actual average number of nonzero links. Thus, I can consider the space complexity for storing the alignment hypotheses to be practically linear in the number of nodes of the trees.

The maximum number of nodes in the trees is also easy to calculate. The biggest tree that can cover a given string would be a full binary tree, which is known to have $(2n - 1)$ nodes, where n is the number of tokens in the string, which in a phrase-structure tree is equal to the number of leaves of the binary tree. If n_S is set to be the number of tokens in the source string and n_T to be the number of tokens in the target string, the maximum total memory used by the system for the storage of the trees will be calculated as in (6).

(6) $2n_S - 1 + 2n_T - 1$

To that I can add the memory needed to store the source and target strings themselves, which is $n_S + n_T$. Following these calculations, the space needed to store the alignment hypotheses is (in practice) roughly equal to the formula in (7) and if I put everything together, I derive the complexity formula in (8). That is, linear space complexity in the number of tokens in the strings, meaning that the system can easily handle very large strings in memory during tree-to-tree alignment.

(7) $2 \cdot \max(n_S, n_T) - 1$

(8) $O(3n_S + 3n_T + 2 \cdot \max(n_S, n_T))$

The situation with the string-to-string alignment case is understandably more complex, as here I am generating all possible binary trees for both the source and target strings. To reduce complexity, this module of the system is implemented to store the generated binary trees as a packed forest in a CYK-style chart (Kasami, 1965, Younger, 1967, Cocke and

Schwartz, 1970), as shown on Figure 21. Such a design is feasible because translational-equivalence scores are based only on the surface strings spanned by the nodes that are being linked, and not on features of the tree structures. Therefore, all nodes that span the same surface string can be represented as a single cell in the chart.

In this way, the memory requirements of the string-to-string algorithm can be easily derived from the sizes of the charts that are being used. Namely, the total number of cell nodes that are being generated is the sum of the number of nodes in the source and target CYK charts and can be seen in (9).

$$\frac{n_S^2 + n_S}{2} + \frac{n_T^2 + n_T}{2} \tag{9}$$

My experiments have shown that the fraction of nonzero alignment hypotheses is even smaller for the string-to-string case, compared to the tree-to-tree case. However, for the sake of simplicity, here I will use the same computation as for the tree-to-tree case. This gives (10) as the practical memory requirement for the storage of the alignment hypotheses. Therefore, when I include the memory requirement for the storage of the surface strings, the complexity formula I obtain is the one in (11).

$$\frac{\left(\max\left(n_S, n_T\right)\right)^2 + \max\left(n_S, n_T\right)}{2} \tag{10}$$

$$O\left(n_S^2 + n_T^2 + 3n_S + 3n_T + \left(\max\left(n_S, n_T\right)\right)^2 + \max\left(n_S, n_T\right)\right) \tag{11}$$

This means that the space complexity here is quadratic in the number of surface tokens. Even though this is significantly worse than for the tree-to-tree case, it is still feasible and allows for the operation on practically all real-life sentence pairs that the system may come across. While it is true that the tree-to-string and string-to-tree modules will use less memory than the string-to-string module, their overall space complexity remains quadratic in the number of the tokens of the sentence, for which a parse is unavailable.

The main difference with respect to space complexity between the greedy-search and the full-search algorithms is that the latter requires significantly more memory for longer sentences. The reason for this is that long sentences (over 40 tokens) might receive hundreds or thousands of different maximal link sets with the same probability. The extra memory in this case would be required to store all such sets that are found during the alignment process. For comparison, the greedy-search algorithm requires the storage of only one final set of links.

3.4.2. Time Complexity

Once I have handled the space complexity of the algorithms, I can turn to the calculation of the time complexity.

To judge the time complexity, I first need to find out what part of the system's operation is most time-consuming. It might not seem intuitive, but the aligner does not do any aligning for the biggest part of its operation. As can be seen on the graph in Figure 23, most of the time is spent calculating the alignment hypotheses' scores for the sentence pair. This behaviour is easy to explain, however. The aligner always has to calculate the translational-equivalence scores for all potential alignment hypotheses. However, most of those hypotheses receive zero scores, as discussed in the previous section. In fact, the longer the sentences that are being aligned, the smaller the fraction of nonzero hypotheses. Therefore, even if all hypotheses were to be converted to actual links, the number of objects the selection algorithm would have to go through would be just a fraction of the number of objects that scores were calculated for.

Having established this, I can now simply consider the calculation of translational-equivalence scores as the main source for time complexity for the alignment algorithm and try to establish what this complexity is. Clearly, it depends on the number of scores that have to be calculated, which in turn depends on the number of nodes in the trees that are being

aligned. Because I are looking at all possible links between nodes in the two trees, the number of potential links will be simply the product of the numbers of nodes in the trees. Using the calculation for the number of nodes from the previous section, for the tree-to-tree case the number of potential links will be as in (12). This gives quadratic time complexity relative to the number of tokens, as seen in (13).

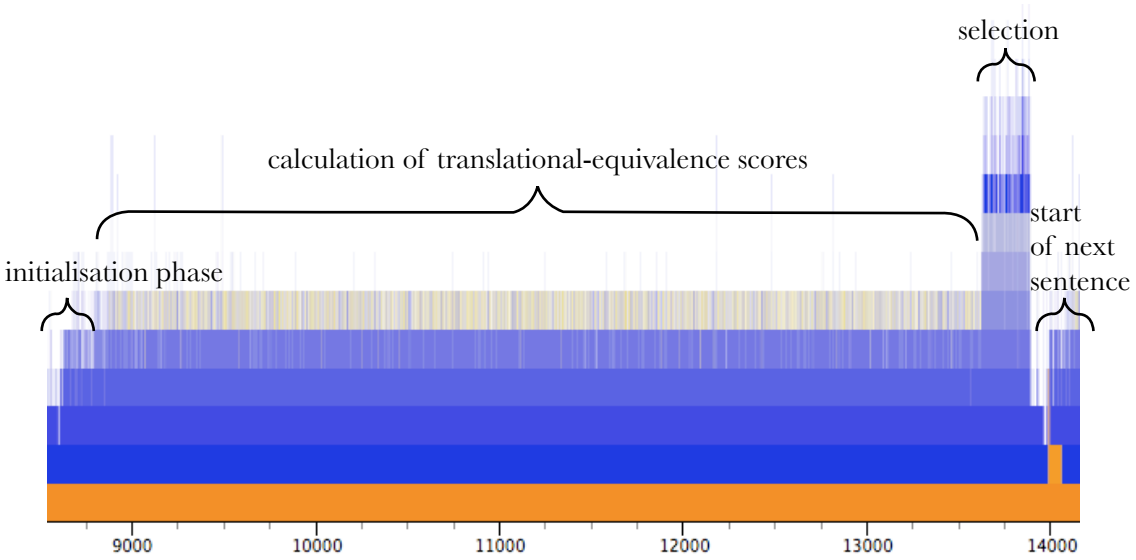


Figure 23: Call-stack graph for the operation of the sub-tree aligner on a single sentence pair, showing the time usage of its different stages

$$(12) \quad (2n_S - 1) \cdot (2n_T - 1)$$

$$(13) \quad O(2n_S n_T - n_S - n_T)$$

As could be expected, the time complexity for the string-to-string case is much higher, since the number of nodes is already quadratic, which results in a much higher number of potential links, viz. (14). Thus, I derive the complexity formula in (15).

$$(14) \quad \frac{n_S^2 + n_S}{2} \cdot \frac{n_T^2 + n_T}{2}$$

$$(15) \quad O(n_S^2 n_T^2 + n_S^2 n_T + n_T^2 n_S + n_S n_T)$$

It can also be seen that a move from string-to-string to tree-to-string or string-to-tree makes a significant difference for the time complexity cal-

culation. Namely, the time complexity for these two cases will only be cubic, rather than polynomial to the fourth power

$$O\left(n_S^2 n_T - \frac{1}{2}n_S^2 + n_S n_T - \frac{1}{2}n_T\right) \quad (16)$$

The calculations presented here suggest that it is highly beneficial to pre-process the data that needs to be aligned with monolingual phrase-structure parsers, if available. The tree-to-string, string-to-tree and especially the string-to-string modules should be used only when it is impossible to find a parser for a particular language or languages.

It should be noted, however, that the score calculation step of the alignment process can be optimised. To see how this can be done, let us consider how the scores are calculated by my algorithm (cf. section 3.1.1). Regardless of the scoring formula that is chosen, the only thing that the translational-equivalence scores depend on are the word-alignment probabilities for aligning the words in the sentence pair. There is no interdependence whatsoever between individual scores. This means that their calculation is a good candidate for parallel execution. Because of this, the software presented in this work has been prepared for the parallelisation of this task. If compiled using a recent version of GCC (4.2 or newer – `http://gcc.gnu.org`), the system can be configured to use all available CPUs (logical and/or physical) on a system to perform the calculation of translational-equivalence scores. Because this operation takes up so much time of the system's operation, even on a dual-core system you can observe a significant reduction in run-time (around 35–40%).

Of course, due to the fact that my system operates on a per sentence pair basis, it also lends itself very well to task-farming parallelisation solutions. However, such solutions are best suited for setups where there are several distinct computation nodes, on each of which a copy of the aligner would be running (worker nodes) and a control node that dispatches sentence pairs to the worker nodes. Given that these computation nodes nowa-

days usually contain multiple CPU cores, the best performing configuration will combine task-farming methods with the built-in parallelisation that I have implemented. In this way the worker-nodes' resources will be best utilised while only keeping one copy of the system in the memory of each node. If, on the other hand, the built-in parallelisation were not present, full utilisation of the worker nodes would require loading as many copies of the system into the memory of each node, as there are CPU cores. It becomes clear why this is problematic, when you consider that each copy of the system will have to load all word-alignment data separately, which for larger data sets amounts to significant overhead.

Something I have not inspected closely so far is the complexity of the selection algorithms themselves. We said earlier that the system spends only a small part of its time performing selection. This, however, only relates to the greedy-search algorithm. The situation with the full-search algorithm is much different, as can be expected. In section 3.1.3, I already mentioned that the full-search algorithm is related to the TSP (Whitney, 1932). The original TSP regards the task of finding the shortest possible tour that visits each city in a list only once, given distances between each two cities. In my case, I am looking for the set of links that produces the highest overall weight. Related to the TSP, the nodes in the trees—each of which can only be linked once—represent the cities and the translational-equivalence scores represent the distances. A major difference to the TSP are the restrictions that I put on the model. Namely, I am only ever linking nodes from the source tree to nodes from the target tree and never nodes within any of the trees. Additionally, the tree structures provide information about incompatibilities between the links, thus limiting the number and form of possible 'tours'. Finally, there are many links that receive zero scores and thus are not considered for linking.

Despite all the restrictions stated above, the problem still remains combinatorial in nature. Therefore, even when using the restrictions to limit the search space by cutting dead-end branches as early as possible, this algorithm can only deal with cases with up to about 200 nonzero alignment hypotheses within a reasonable timeframe. Because of this, I have used this approach only to validate the performance quality of the greedy-search-based selection algorithm on the HomeCentre data set (described in section 4.1). Ignoring the selection phase, this, as will be discussed in more detail in section 4.1.3, has shown that the restrictions imposed on the model allow the discovery of the best solution in up to 95% of the cases considered with only polynomial time complexity (the order of the polynomial depending on the operation mode of the system, as discussed above), which is not possible for the generalised version of the TSP.

3.5. Conclusions

In this chapter I presented the underlying design of my system for automatic generation of parallel treebanks. The software can be configured to use either a fast robust greedy-search-based selection algorithm, or an exhaustive full-search-based one. Both algorithms can use one of two available scoring functions: *score1* or *score2*.

For the greedy-search algorithm, two algorithms for dealing with ambiguities were developed—*skip1* and *skip2*—allowing it to deal with complex alignment situations. Additionally, the extension *span1* was developed to help mitigate possible weaknesses of the scoring functions.

As an attempt to allow the greedy-search algorithm to self-adjust its operation, a re-scoring mechanism was designed. It ensures that each newly generated link for a tree pair affects the choice of subsequent links in the selection process.

Along with the basic *tree-to-tree* alignment module, *string-to-string*, *tree-to-string* and *string-to-tree* modules were developed. This ensures the maximum utilisation of any available resources for the language pair in question: monolingual phrase-structure parsers or Part-of-Speech taggers. It also allows for the generation of parallel treebanks for pairs of under-resourced languages, for which very few external resources exist.

Considering the space and time complexity of the employed algorithms, I regard the system presented in this work to be a useful tool that can operate on very large parallel corpora to produce parallel treebanks. Although a parallel treebank can be generated in a completely unsupervised manner—given a parallel text—using only existing open source tools (eg. GIZA++ – Och and Ney, 2003) and the software presented in this work, the use of parsers to pre-process the data is recommended and beneficial, as it can significantly reduce the time required for sub-tree alignment. It should be noted, however, that the generation of parallel treebanks is not a time-critical task as there are no known applications that require on-the-fly generation.

Even though I designed a full-search algorithm that always produces the best set of alignments for a given tree pair, I showed that its use is limited to small data sets and evaluation tasks, due to its relatedness to the Travelling Salesman Problem, which makes it combinatorial in nature and not computationally feasible for real-life tasks. However, it can be used on small scale to assess the quality and appropriateness of the greedy-search-based algorithm.

In the next chapter, I present the battery of tests that I developed for the intrinsic and extrinsic evaluation of the system described in this work. The evaluation results showcase the various features of my software and point out the high quality and usability of the generated parallel treebanks.

4. Evaluation and Testing

In the previous chapter, I presented my platform for the automatic generation of parallel treebanks. I had a detailed look at the various possible configurations and operation modes. Now I will put the system to the test by means of a battery of tests that I developed, using both intrinsic and extrinsic evaluations.

The quality of a parallel treebank depends directly on the quality of the sub-tree alignments that it contains. Because of this, I use the evaluation results mainly as a metric for the performance of the underlying sub-tree alignment system during development. Of course, the evaluation presented in this chapter also presents an insight into the usability of the parallel treebanks produced using my method.

As a reference for the tests, a hand-crafted parallel treebank was used (HomeCentre – Hearne and Way, 2006). This treebank consists of 810 English–French sentence pairs from a Xerox printer manual. Both the source and target side were parsed manually at Xerox PARC. The sub-tree alignments were introduced manually by a single annotator at DCU (Hearne, 2005). As discussed in section 2.1, I am not aware of an existing parallel treebank besides the HomeCentre that can be used directly for cross-evaluation and comparison to versions automatically generated using the sub-tree aligner.

The word-alignment probabilities required by my system are based on IBM Model 4 word alignments on the plain sentences of the HomeCentre in both language directions. These were obtained with the use of the training scripts supplied with the Moses decoder (Koehn et al., 2007). In order to run the *string-to-string*, *string-to-tree* and *tree-to-string* modules, I used a version of the HomeCentre with no parse trees. The original POS-tags were left intact, however.

In addition to the HomeCentre treebank, I used a treebank consisting of 10,000 sentence pairs from the English–German part of the EuroParl (Koehn, 2005). The English side was parsed using Bikel's parser (2002) trained on the Penn Treebank (Marcus et al., 1993), and the German side was parsed using BitPar (Schmid, 2004) trained on the TIGER treebank (Brants et al., 2002). The parse trees were not altered in any way after parsing. The word-alignment probabilities and the POS-tagged version of this data set were obtained as for the HomeCentre.

Further in the text I would take the English side in both data sets to be the source side, while the French side of the HomeCentre will be referred to as the target side, as will be the German side of the EuroParl.

There are several crucial differences between the HomeCentre and EuroParl data sets I used. First, the EuroParl contains automatically generated parses, while the HomeCentre has been parsed manually. Therefore the HomeCentre has far superior parse quality to the EuroParl data set I used. Also, the HomeCentre contains shorter sentences on average, compared to the EuroParl, cf. Table 4. Finally, I have a manually sub-sententially aligned version only of the HomeCentre. Manual annotation of the EuroParl data would take a prohibitively long time and so was not undertaken.

Data Set	HomeCentre		EuroParl	
Language	English	French	English	German
Tokens	8.83	10.05	17.99	16.83

Table 4: Average sentence length for
the HomeCentre and EuroParl data sets

Besides the alignments produced using our system and the manual alignments for the HomeCentre, I also obtained a set of alignments produced using the *TnT* training algorithm designed for the Stat-XFER system (Ambati and Lavie, 2008).[7]

[7] I would like to thank Vamshi Ambati, Jonathan Clark and Alon Lavie from Carnegie Mellon University for their help and co-operation with the generation of this data set.

Configuration	Code
skip1, *score1*	11
skip1, *score1*, *span1*	111
skip1, *score2*	12
skip1, *score2*, *span1*	121
skip2, *score1*	21
skip2, *score1*, *span1*	211
skip2, *score2*	22
skip2, *score2*, *span1*	221
skip1, *score1*, *rescore*	11r
skip1, *score1*, *span1*, *rescore*	111r
skip1, *score2*, *rescore*	12r
skip1, *score2*, *span1*, *rescore*	121r
skip2, *score1*, *rescore*	21r
skip2, *score1*, *span1*, *rescore*	211r
skip2, *score2*, *rescore*	22r
skip2, *score2*, *span1*, *rescore*	221r

Operation Mode	Code
String-to-String	str2str
String-to-Tree	str2tree
Tree-to-String	tree2str
Tree-to-Tree	tree2tree
Tree-to-Tree	tree2tree

Search Mode	Code
Greedy Search	gs
Full Search	fs

Table 5: Abbreviation codes used to refer to the configurations and operation modes of the sub-tree aligner

Throughout this chapter I will use the codes from Table 5 to designate the various possible configurations and operation modes of the sub-tree aligner. The labels in the *Configuration* column refer to the type of *skip* module and scoring function that are being used and whether the *span1* module and/or the *rescore* modules are turned on. For example, *tree2tree_121_gs* denotes the use of the *tree-to-tree* alignment module employing greedy search (*gs*), setup with the *skip1* module, the *score2* scoring function and the *span1* module. Additionally, I will use the code *man_gold* to refer to the manually aligned version of the HomeCentre and *CMU_TnT* for the data set supplied by the team at CMU.

It should be noted that the full-search-based selection algorithm supersedes the use of either *skip* module and that a version of the *span1* module was not implemented for the full search algorithm. Also, due to the time requirements of the full-search algorithm, I only used it within the tree-to-tree module for both *score1* and *score2*. More details about the setup of the experiments employing the full-search algorithm can be found in section 4.1.3.

A complete description of the operation and properties of all configurations of my aligner is given in Chapter 3. An overview of the algorithms used to obtain the *CMU_TnT* data set is presented in section 2.2.2.

I will first describe the intrinsic testing in section 4.1, which consists of an in-depth analysis of the properties of the automatically generated data sets in section 4.1.1 and evaluation against the gold-standard data set in section 4.1.2. The intrinsic evaluation also includes assessing the quality of the data sets produced using the greedy-search-based selection algorithm against the performance of the full-search-based selection in section 4.1.3.

Then I will go into the details of the extrinsic evaluation in section 4.2. For this evaluation I use an existing MT system (DOT – Hearne and Way, 2006) to perform translation experiments by training the system on both the manually and automatically generated data sets. I then use the quality of the produced translation as the metric representing the quality of the parallel treebanks used for training. Finally, in section 4.3 I will conclude the chapter.

4.1. Intrinsic Evaluation

I will start the intrinsic evaluation by looking at the properties of the automatically generated parallel treebanks in comparison to the properties of the gold-standard parallel treebank in section 4.1.1. Afterwards in section 4.1.2, I will evaluate the generated data sets against the gold standard by comparing the links induced by the automatic aligners to the manually annotated links in the HomeCentre treebank. This evaluation can only be performed for the *CMU_TnT* data set and for the parallel treebanks produced by the tree-to-tree alignment module of our aligner, as the string-to-string, string-to-tree and tree-to-string modules generate new trees for at least one of the language sides. Additionally, I evaluate the performance of our system against the *CMU_TnT* data.

For the evaluation outlined above, I only use data sets generated using the greedy-search-based algorithm of my aligner. I used the versions of the HomeCentre treebank that I generated using the *tree-to-tree* module and the full-search algorithm as a benchmark for the quality of the greedy-search algorithm. The details are presented in section 4.1.3.

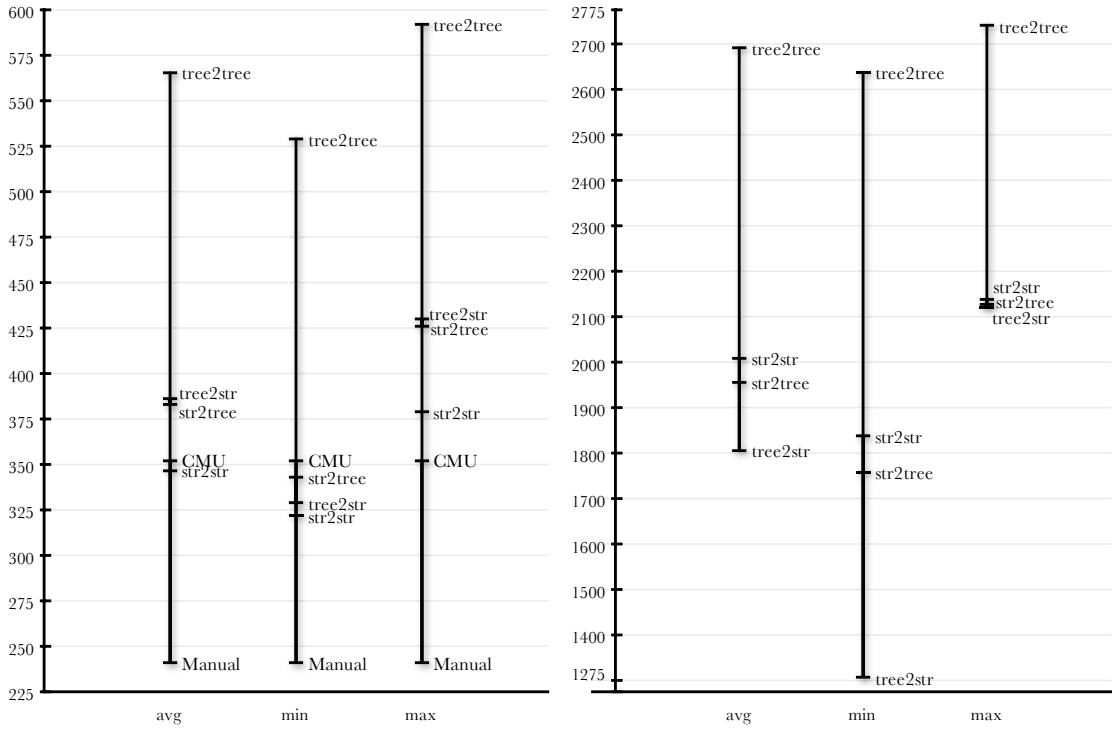
4.1.1. Analysing the Parallel Treebanks

I start with a look at some of the characteristic properties of the various parallel treebanks I have available. Based on this analysis, I make some predictions about the performance of the different configurations of my aligner. I will use the various experiments detailed later in this chapter to find out whether these predictions were borne out and if not, I will try to give convincing explanations.

As was mentioned earlier, one of the major differences between the HomeCentre and EuroParl data sets is the average sentence length. It can be seen from the data in Table 4 on p.59 that EuroParl contains sentences that are on average over twice as long as the sentences in the HomeCentre (looking at the English side).

In Tables 1 – 3 in Appendix A, you can find the number of node labels on the source and target sides, as well as the number of unique linked node-label pairs resulting from the sub-sentential alignment. An overview of the distribution of the POS tag-pairs is given in Figure 24. The parallel treebanks produced using the *tree-to-tree* module, as well as the *CMU_TnT* and *man_gold* data sets, use the full sets of node labels available in the original parsed data, while the data produced using the *string-to-string*, *string-to-tree* and *tree-to-string* modules only use the POS tags from the original data plus the special *X* node label that is assigned to the automatically generated nodes on the language side(s) for which the input was not parsed. This explains the fact that the configurations of my aligner using the *tree-to-tree* module produce the highest number of node-label pairs for the HomeCen-

tre, while the ones using the *string-to-string* modules produce the least number of node-label pairs and the other modules fall in the middle. The *CMU_TnT* data set has about as many unique node-label pairs as our *string-to-string* data, even though the algorithm is comparable to the *tree-to-tree* module (cf. section 2.2.2), and the *man_gold* data set has by far the least number of node-label pairs.



HomeCentre POS tag-pairs EuroParl POS tag-pairs

Figure 24: Average, minimum and maximum number of unique source ↔ target pairs of linked Part-of-Speech tags in the HomeCentre and EuroParl data sets

There are two possible explanations for these results. Either the sub-sentential links within the *CMU_TnT* and *man_gold* data sets are more consistent than the ones produced by my aligner, or there are fewer links; the exact reason will be investigated later in this section. An interesting fact is that the configurations employing the *rescore* module generally have fewer unique node-label pairs compared to their non-*rescore* counterparts.

These statistics differ, however, for the EuroParl data set in that the *string-to-tree* and *tree-to-string* modules produce the smallest amount of node-label pairs,

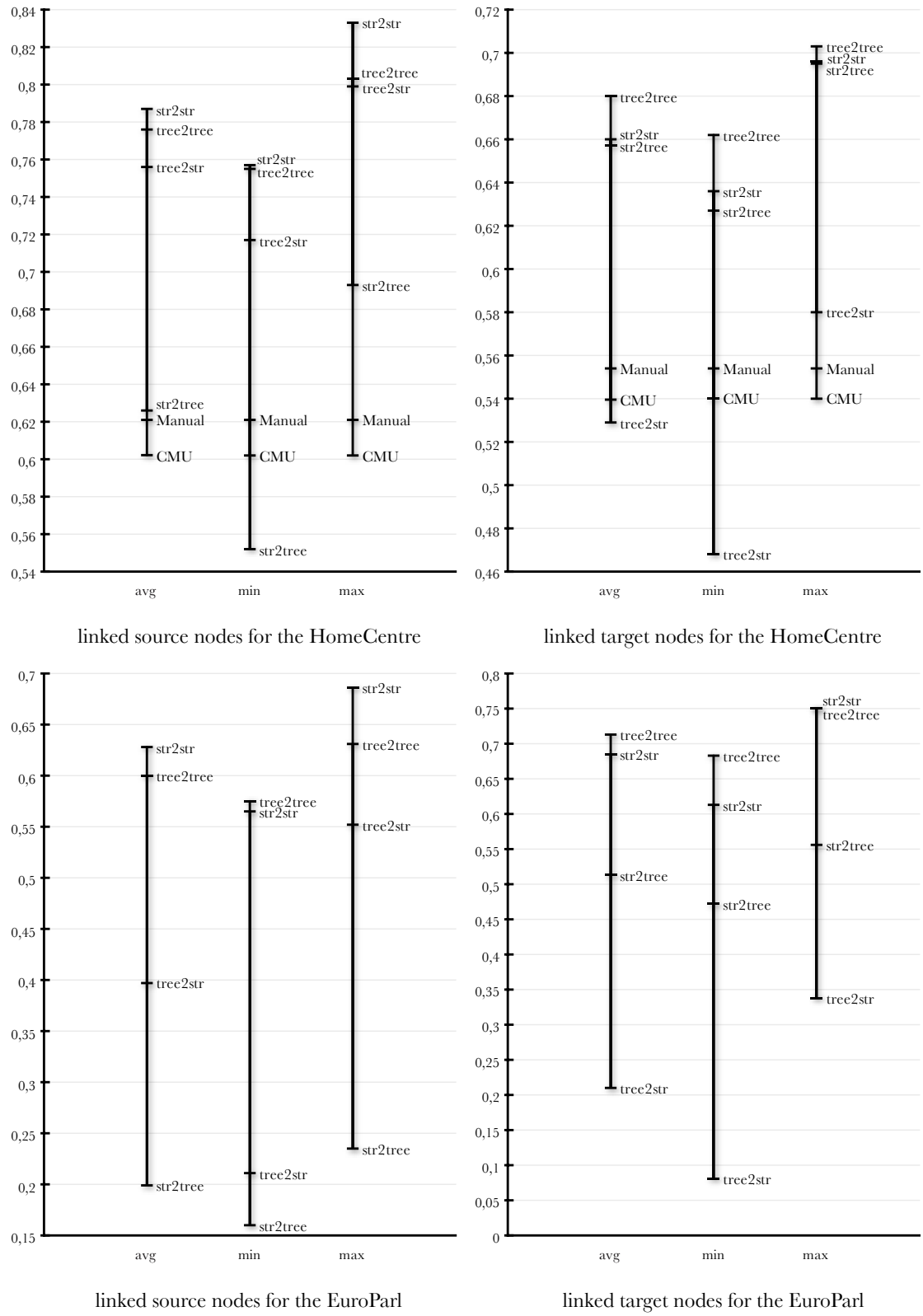


Figure 25: Average percentage of linked nodes in source and target trees in the HomeCentre and EuroParl data sets

rather than the *string-to-string* module as for the HomeCentre data. Again, this can either be due to higher consistency among the alignments or due to

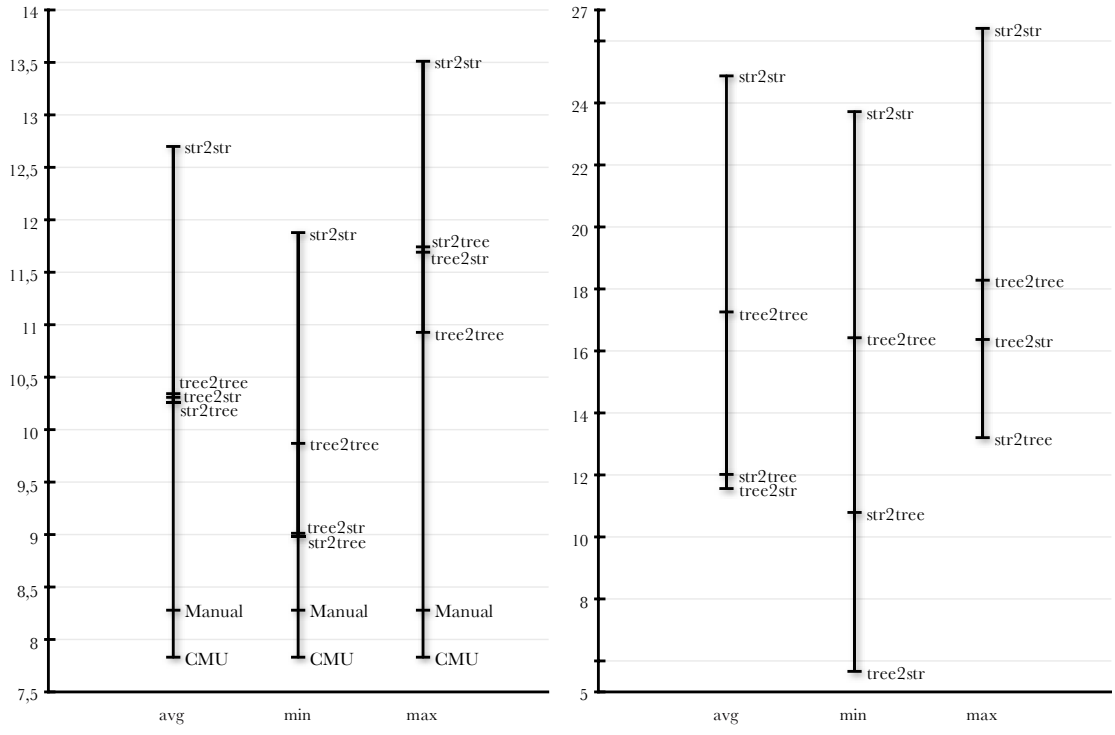
a smaller number of links and the actual reason will be investigated during the further analysis of the data in this section.

Turning to Table 4 from Appendix A for the HomeCentre (summarised in Figure 25), you see that the *string-to-string* module has the highest percentage of linked nodes on average for both source and target trees. This is due to the fact that the phrase structure trees here are motivated solely by the existence of translationally equivalent strings in both languages and thus they are generally the minimal trees that accommodate the available translational-equivalence data. This can also be observed for the EuroParl data set in Figure 25. The statistics for the EuroParl data actually make it even clearer that the *string-to-tree* and *tree-to-string* modules produce the smallest amount of linked nodes on average. You will see some possible reasons for this later, but at this stage I can say that this fact has to be related to the reduced number of node-label pairs in the data sets produced by these modules.

The data in Tables 4 and 5 in conjunction with the data in Tables 6 and 7 (all from Appendix A) also give information about the depth of the phrase-structure trees in the various parallel treebanks. One might argue that the *string-to-string* module would produce the best structures that can accommodate the translational equivalencies between the two languages in question, as it is not restricted by any existing structures that might be suboptimal. Therefore, based on the number of nodes produced by the *string-to-string* module, I can conclude that the original trees in the HomeCentre treebank could be flatter—especially on the target side—while the trees produced by the parsers for the EuroParl data set have optimal depth on the source side and need to be considerably deeper on the target side.

Both for the HomeCentre and EuroParl data sets I find that the *string-to-tree* and *tree-to-string* modules produce considerably fewer nodes on the language side where plain input is used, compared to the *string-to-string*

module and to the original parse trees. I believe that this can be linked to the data in Tables 8 and 9 from Appendix A, namely the number of links produced by the different configurations. A summary of this data is shown on Figure 26. You see that—particularly for the EuroParl data—these two modules produce the smallest number of links. A plausible explanation for this finding lies in the fact that different natural languages can rarely be analysed using the same structure. The *string-to-tree* and *tree-to-string* modules, however, are forced to use the structures defined for one language to constrain and guide their decisions when generating structure for the other language. The data seem to point out that—when trying to capture the translational equivalencies in two languages in a structural manner—one should either start building structure for both languages from scratch, or use existing structures that have been specifically tailored to fit the languages in question.



avg number of links for the HomeCentre avg number of links for the EuroParl

Figure 26: Average number of sub-sentential alignments in the HomeCentre and EuroParl data sets

An interesting observation is that all configurations of my system produce (considerably) more sub-sentential links than exist in the *man_gold* data and much more than the links in the *CMU_TnT* data. I can draw the following conclusions from this. The manual annotation of parallel treebanks is a very difficult and tedious precision-oriented process (Hearne et al., 2007, Samuelsson and Volk, 2007) and it is easy for the annotator to omit proper alignments when s/he is unsure whether to link certain nodes or not. This is true in particular of the HomeCentre parallel treebank which was annotated by a single person.

As regards the *CMU_TnT* data, further analysis was needed to find out what was responsible for the small number of links. I discovered that, with respect to the number of tokens covered by linked nodes, the *CMU_TnT* data only contains one-to-one and many-to-many links. We believe that the lack of one-to-many and many-to-one links of this type is severely limiting this data set, as often what can be expressed with a single word in one language, has to be expressed using several words in another. If you also look at Table 10 from Appendix A, you will see that the distribution between non-lexical and lexical links for this data set is comparable only to the data produced by the *string-to-string* module. What these statistics, together with the data about the average number of links, show is that the CMU algorithm produces significantly fewer lexical alignments compared to the system presented in this work. In fact, only the *man_gold* data set has fewer lexical links on average than the *CMU_TnT* data set.

I believe that it is crucial for a parallel treebank to have a high number of lexical links, as they constitute the fundamental building blocks that can be used to represent translational equivalence. Later on in this chapter I will discuss whether and to what extent the need for more lexical links affects the usefulness of these data sets within a machine translation setting. The statistics presented so far leave the impression that the CMU algorithm

is inferior to the sub-tree alignment system presented in this work, mostly due to the smaller number of alignments—and especially lexical alignments—and due to the lack of one-to-many alignments with regard to the number of tokens covered by a particular link.

In Tables 12 and 13 in Appendix A, you see the total number of sentences for which the automatic aligner could not resolve all available ambiguities per configuration, as well as the average number of ambiguous links for these sentences. Tables 14 and 15 in the same appendix present the average number of search space reductions per sentence pair per configuration, i.e. the number of iterations that the greedy-search-based algorithm had to go through before exhausting the list of alignment hypotheses available for linking.

For the HomeCentre data set, Tables 12 and 14 do not show significant variations, which I believe is due to the small size of the treebank. For the EuroParl data in Tables 13 and 15, on the other hand, I see a clear trend for the *tree-to-tree* module to encounter a larger number of ambiguous cases. This can easily be explained by the fact that the other alignment modules have more freedom to select structural links, and are thus less likely to come across irresolvable ambiguities. Conversely, the *string-to-string* module has to perform the greatest number of steps while doing alignment. This can be directly attributed to the much higher number of hypothetical links it has to operate on.

Interestingly, the *string-to-tree* and *tree-to-string* modules require the fewest number of steps on average during alignment, even though they operate on more alignment hypotheses that the *tree-to-tree* alignment module. I believe that this is because the system would try to use the existing phrase structure on one language side as much as possible for disambiguation and for the blocking of unwanted hypotheses.

With this I conclude the analysis of the properties of the various data sets. In the next section, I will discuss to what extent the automatic sub-tree aligners are able to match the gold standard I have for the HomeCentre. Presently, I will focus only on this data set and I will return to the EuroParl in section 4.2.

4.1.2. Evaluation Against the Gold Standard

Now I can turn to the evaluation of the automatically generated parallel treebanks against the gold standard. The metrics I use are precision and recall for all sub-tree alignments and lexical and non-lexical alignments alone. The full results of the evaluation are shown in Table 6 and Figure 27.

System/ Configuration	All links		Lexical links		Non-lexical links	
	precision	recall	precision	recall	precision	recall
CMU_TnT	72.25%	70.85%	66.00%	70.77%	83.18%	72.04%
tree2tree_11_gs	60.10%	75.87%	49.64%	79.55%	83.13%	72.37%
tree2tree_11r_gs	59.94%	73.62%	49.39%	76.87%	83.83%	70.17%
tree2tree_111_gs	61.31%	79.27%	51.06%	81.40%	79.38%	77.90%
tree2tree_111r_gs	61.96%	77.77%	51.53%	79.39%	80.18%	76.73%
tree2tree_12_gs	61.29%	77.46%	51.06%	79.99%	80.75%	75.69%
tree2tree_12r_gs	61.66%	75.77%	51.48%	78.48%	80.93%	73.88%
tree2tree_121_gs	61.56%	78.44%	51.53%	80.51%	78.67%	77.22%
tree2tree_121r_gs	62.48%	77.62%	52.50%	80.06%	79.37%	76.34%
tree2tree_21_gs	60.80%	76.27%	50.12%	79.71%	82.82%	73.01%
tree2tree_21r_gs	60.30%	73.81%	49.68%	77.12%	83.40%	70.32%
tree2tree_211_gs	61.56%	79.25%	51.19%	81.35%	79.38%	77.91%
tree2tree_211r_gs	62.13%	77.80%	51.58%	79.40%	80.22%	76.77%
tree2tree_22_gs	61.54%	77.50%	51.29%	80.03%	80.75%	75.70%
tree2tree_22r_gs	61.82%	75.75%	51.64%	78.45%	80.93%	73.88%
tree2tree_221_gs	61.79%	78.49%	51.76%	80.60%	78.73%	77.22%
tree2tree_221r_gs	62.64%	77.60%	52.66%	80.02%	79.40%	76.34%

Table 6: Intrinsic evaluation of the automatically aligned versions of the HomeCentre parallel treebank against the gold standard

As I said earlier, this evaluation can only be performed for the output of the *tree-to-tree* alignment module, as all other modules produce new struc-

tures on at least one language side, which cannot be directly compared to the gold-standard trees. I will start by evaluating the parallel treebanks produced by our system and the *CMU_TnT* data set against the gold standard. Then I will use the same techniques to evaluate the data produced by my aligner against the *CMU_TnT* data in a bid to better understand where the two approaches differ, and where they produce comparable results.

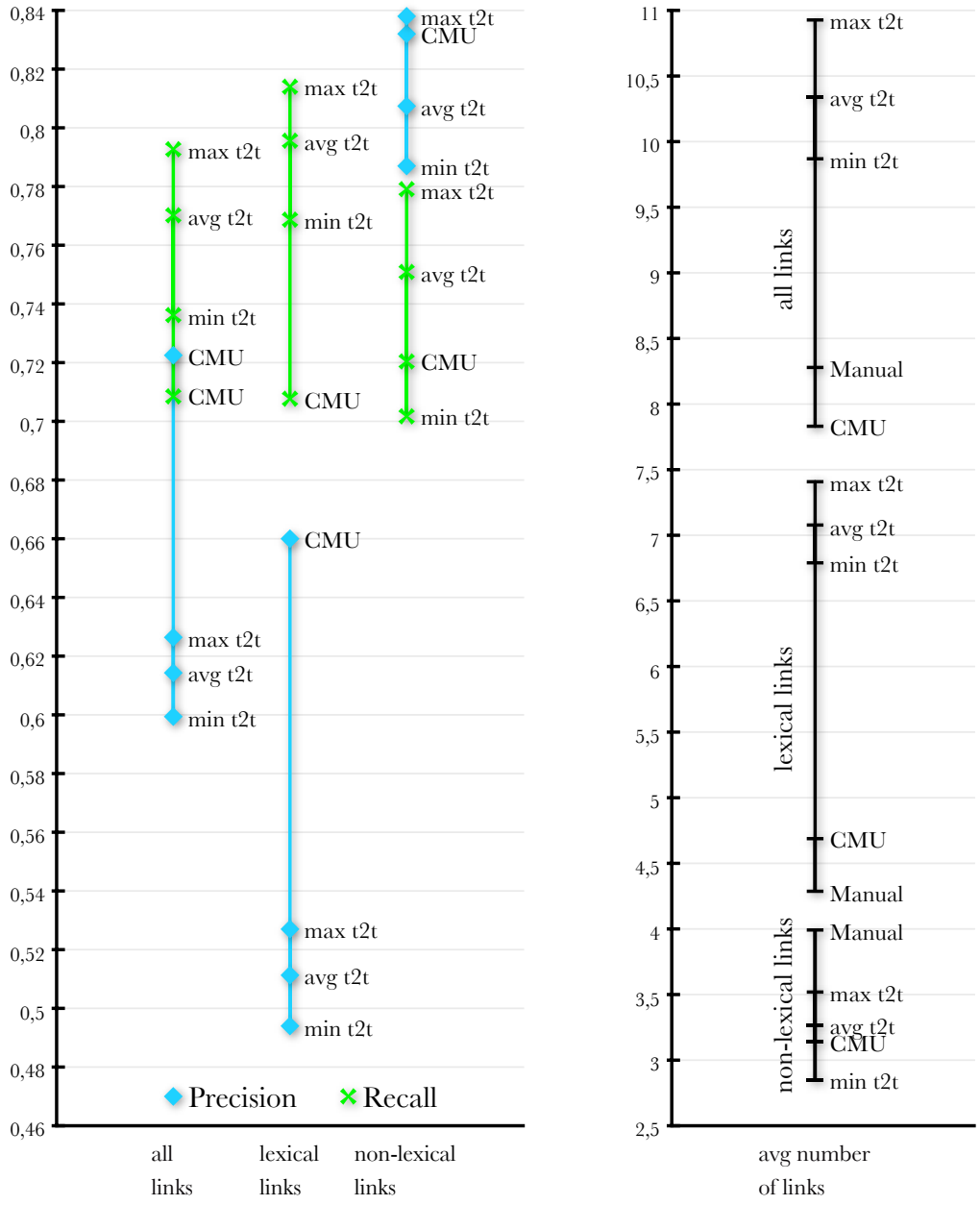


Figure 27: Comparison between the precision and recall of the *tree-to-tree* module and the CMU aligner evaluated against the gold standard for the HomeCentre parallel treebank, as well as a comparison between the number of links present in the various data sets

The first observation that can be made when looking at the *All links* column of Table 6 and at the left-hand graph from Figure 27 is that for all configurations of my aligner the precision is significantly lower than the recall. This gap becomes even bigger when we concentrate on the lexical links, where the average precision is around 51% while the average recall is about 79%. At the same time, the data for the non-lexical links show consistently higher precision than recall. On the other hand, the *CMU_TnT* data shows a better balance between precision and recall with the precision in the *All links* column being higher than the recall.

To understand the cause of these results, I will have to go to Tables 8 and 10 in Appendix A and to the right-hand graph on Figure 27. As was seen earlier, the gold-standard tree pairs have fewer links on average than the tree pairs produced by any configuration of our sub-tree aligner. This is especially true for the lexical links, as the *man_gold* data shows a stronger preference for non-lexical links than the automatically generated data sets. Therefore, I can conclude that the lack of precision is due to the production of additional lexical links that do not occur in the gold standard, rather than to not being able to select the proper lexical alignments, which is supported by the high recall values. On the other hand, it seems that the automatic aligner cannot discover all available non-lexical alignments, hence the drop in recall there. What these data suggest is that the aligner needs to produce a larger number of non-lexical links and significantly fewer lexical ones if it was to better match the gold standard. However, I do not currently know whether better matching the gold standard will necessarily result in parallel treebanks that are better suited for use in machine translation tasks. Therefore, I will use the extrinsic evaluation techniques in section 4.2 to assess whether I indeed need to strive to reach the gold standard or not.

For the *CMU_TnT* data set, I observe similar precision and recall to our aligner for the non-lexical links. On the other hand, this data set shows significantly higher precision for the lexical links, but also a drop in recall.

These results might have been explained by the lower number of links present in the *CMU_TnT* data, compared to the gold standard, but a more careful examination of the data show that there are on average 4.69 lexical links in the *CMU_TnT* data set versus 4.29 for the gold standard. Therefore, there is a significant difference between the two sets of lexical links and the CMU algorithm has problems finding the proper word-level alignments. I believe that this is due to its over-reliance on the lexical alignments produced by GIZA++; as discussed in section 2.2.2, the CMU algorithm uses the GIZA++ alignments for each sentence pair directly during the alignment process, while my algorithm uses a probabilistic bilingual dictionary derived from the word-alignment data instead (cf. section 3.1).

System/ Configuration	All links		Lexical links		Non-lexical links	
	precision	recall	precision	recall	precision	recall
tree2tree_11_gs	58.40%	75.68%	52.76%	79.18%	73.12%	76.92%
tree2tree_11r_gs	58.20%	73.36%	52.40%	76.76%	73.94%	74.59%
tree2tree_111_gs	59.74%	79.40%	54.77%	81.59%	72.17%	83.37%
tree2tree_111r_gs	60.29%	77.76%	55.18%	79.35%	72.79%	82.43%
tree2tree_12_gs	59.07%	76.82%	53.94%	79.35%	72.62%	80.54%
tree2tree_12r_gs	59.57%	75.17%	54.62%	77.69%	72.74%	78.80%
tree2tree_121_gs	59.43%	77.83%	54.79%	80.24%	70.90%	82.04%
tree2tree_121r_gs	60.40%	76.96%	55.92%	79.26%	71.41%	81.17%
tree2tree_21_gs	58.95%	76.09%	53.35%	79.38%	73.40%	77.77%
tree2tree_21r_gs	58.45%	73.46%	52.66%	76.77%	74.05%	74.96%
tree2tree_211_gs	59.89%	79.33%	54.92%	81.46%	72.18%	83.40%
tree2tree_211r_gs	60.30%	77.68%	55.14%	79.20%	72.86%	82.57%
tree2tree_22_gs	59.32%	76.84%	54.22%	79.37%	72.64%	80.58%
tree2tree_22r_gs	59.72%	75.14%	54.82%	77.64%	72.74%	78.80%
tree2tree_221_gs	59.67%	77.89%	55.07%	80.32%	70.92%	82.04%
tree2tree_221r_gs	60.55%	76.93%	56.12%	79.20%	71.44%	81.17%

Table 7: Intrinsic evaluation of versions of the HomeCentre parallel treebank aligned using our aligner against a version aligned using the CMU aligner

I also used the precision and recall metrics to compare the parallel treebanks produced using my method against the *CMU_TnT* parallel tree-

bank in an attempt to better understand the similarities and differences in the operation of the two systems. The results are presented in Table 7.

Immediately can be seen that, as with the evaluation against the gold standard, precision is significantly lower than recall. Here again, I attribute this to the higher number of lexical links produced by my algorithm. However, there is higher coincidence of the lexical links in the output of our system with the lexical links in the *CMU_TnT* data set, than with the lexical links in the gold standard. This is a direct result of both automatic systems using the same word-alignment data produced by GIZA++. Nonetheless, I do not observe an exact match because the CMU system directly replicates and filters the one-to-one lexical alignments produced by GIZA++ for each particular sentence pair and excludes any one-to-many alignments; meanwhile my system uses a probabilistic bilingual dictionary derived from the GIZA++ word alignments, thus being able to side-step errors present in the original word-alignment data and to find new possible alignments that GIZA++ had skipped for the particular sentence pair.

There is also a mismatch in the non-lexical alignments that the two automatic systems produce. My algorithm finds on average 80% of the non-lexical links produced by the CMU algorithm, plus some additional non-lexical links. However, there are links that the CMU system produces and mine does not. I believe that this is a direct result of the mismatch on the lexical level and not a separate phenomenon.

In this section, I saw that my automatic sub-tree aligner produces the majority of the links present in the gold-standard data set, as attested by the high recall statistics. However, the system produces a significantly higher number of links—in particular lexical links—which leads to a low precision overall. The data sets produced by my aligner relate to the *CMU_TnT* data set similarly to the way they relate to the gold standard. However, the *CMU_TnT* differs significantly from the *man_gold* one. Therefore, I cannot be

confident about which automatic alignment algorithm performs better overall before I have looked at the results of the extrinsic evaluation in section 4.2.

The important point that needs to be taken from this section is that the automatic algorithm should produce more non-lexical links and fewer lexical ones, if it were to match the gold standard. However, I am confident that the extra lexical links being produced are beneficial in the machine translation context and therefore I am uncertain that matching the gold standard will necessarily produce the best results. It will be seen whether this is indeed the case in section 4.2.

4.1.3. The Full-Search-Based Algorithm as a Benchmark

As discussed in section 3.4, the full-search selection algorithm is combinatorial in nature and, hence, for sentence pairs with more than 100 nonzero link hypotheses, its time requirements become prohibitively expensive. Nevertheless, this algorithm can be used in its current form for development purposes.

It is reasonable to ask whether the greedy-search algorithm produces the best set of alignments for a given sentence pair. It could be that it picks a local maximum differing greatly from the absolute maximal set of alignments, thus producing either low-quality links or a small number of links.

The full-search selection algorithm can be used to test the performance of the greedy-search algorithm, as it by definition produces the best available set of alignments for a given tree pair. I decided to use the rate of coincidence between the alignments induced using both selection algorithms as a metric for the quality of the links derived using the greedy search: the higher the number of cases in which the greedy-search algorithm matches the result of the full-search algorithm, the better the quality of the greedy search.

Tables 8 and 9 show two sides of this evaluation. In Table 8, I present the percentage of sentences for which the various configurations of the aligner with the greedy-search-based algorithm matched exactly the set of links determined by the full-search-based algorithm. This coincidence rate is presented both as a total and separately for sentences for which the greedy-search could not resolve all ambiguities, as well as for sentences for which all ambiguities were successfully resolved. In Table 9, I look at the precision and recall of the links produced by the greedy-search algorithm, given the full-search-based alignments as a standard, in the same manner used for the evaluation presented in section 4.1.2.

System/ Configuration	Total solutions found	Sentences with undecided links		Sentences without undecided links	
		#	% found	#	% found
tree2tree_11_gs	87.95%	76	68.42%	704	90.06%
tree2tree_11r_gs	62.44%	72	30.56%	708	65.68%
tree2tree_111_gs	77.18%	66	68.18%	714	78.01%
tree2tree_111r_gs	58.21%	61	31.15%	719	60.50%
tree2tree_12_gs	95.64%	82	91.46%	698	96.13%
tree2tree_12r_gs	60.64%	86	34.88%	694	63.83%
tree2tree_121_gs	83.72%	84	79.76%	696	84.20%
tree2tree_121r_gs	57.69%	78	37.18%	702	59.97%
tree2tree_21_gs	89.62%	129	73.64%	651	92.78%
tree2tree_21r_gs	62.69%	101	27.72%	679	67.89%
tree2tree_211_gs	77.18%	99	72.73%	681	77.83%
tree2tree_211r_gs	58.21%	80	28.75%	700	61.57%
tree2tree_22_gs	95.90%	107	92.52%	673	96.43%
tree2tree_22r_gs	60.51%	103	35.92%	677	64.25%
tree2tree_221_gs	84.23%	105	82.86%	675	84.44%
tree2tree_221r_gs	57.69%	94	38.30%	686	60.35%

Table 8: Coincidence rate for the greedy-search-based algorithm in exactly matching the alignments produced by the full-search-based algorithm for the HomeCentre

The present evaluation was performed by initially producing two parallel treebanks by running full-search-based alignment on the HomeCentre data set first using *score1* and then using *score2* as the scoring func-

tion. The evaluation was performed only for the *tree-to-tree* module, as the other alignment modules have a significantly higher initial number of nonzero alignment hypotheses. In order to be able to perform this evaluation within a reasonable timeframe, I limited the number of nonzero hypotheses to 100, i.e. I skipped sentence pairs for which there were more than 100 initial nonzero hypotheses. For the HomeCentre data set this excluded just 30 of the 810 sentence pairs. Still, the alignment took about 70 minutes, compared to the 4 seconds it takes the greedy-search-based algorithm to align all 810 sentence pairs.

System/ Configuration	All links		Lexical links		Non-lexical links	
	precision	recall	precision	recall	precision	recall
tree2tree_11_gs	95.94%	97.13%	95.16%	97.84%	98.53%	95.13%
tree2tree_11r_gs	93.80%	92.12%	92.66%	92.72%	97.41%	90.39%
tree2tree_111_gs	93.56%	96.81%	94.37%	96.12%	91.49%	98.39%
tree2tree_111r_gs	92.33%	92.50%	91.96%	90.43%	92.68%	97.18%
tree2tree_12_gs	98.71%	99.18%	98.36%	99.14%	99.57%	99.21%
tree2tree_12r_gs	95.85%	93.21%	94.50%	91.72%	98.79%	95.83%
tree2tree_121_gs	95.78%	97.00%	95.92%	96.33%	95.19%	99.21%
tree2tree_121r_gs	93.71%	92.11%	92.72%	89.79%	95.32%	97.51%
tree2tree_21_gs	96.85%	97.52%	96.36%	98.12%	98.08%	95.83%
tree2tree_21r_gs	94.25%	92.25%	93.27%	92.83%	97.13%	90.59%
tree2tree_211_gs	93.98%	96.83%	94.88%	96.14%	91.50%	98.42%
tree2tree_211r_gs	92.55%	92.54%	92.24%	90.46%	92.71%	97.28%
tree2tree_22_gs	99.23%	99.21%	99.06%	99.17%	99.57%	99.23%
tree2tree_22r_gs	96.14%	93.16%	94.88%	91.64%	98.79%	95.83%
tree2tree_221_gs	96.27%	97.09%	96.58%	96.44%	95.25%	99.21%
tree2tree_221r_gs	94.04%	92.09%	93.16%	89.75%	95.36%	97.51%

Table 9: Intrinsic evaluation of versions of the HomeCentre parallel treebank aligned using the greedy-search-based algorithm against their counterparts aligned using full-search-based algorithm

The first impression from looking at Table 9 is that both the precision and recall scores are very high—over 92%—which is very encouraging. The results in Table 8, however, are worse. Nevertheless, there are several trends that are consistent in both tables.

First, it can be noticed that the worst results overall (by a significant margin in Table 8) are obtained from configurations including the *rescore* module. This is surprising, as I had expected the dependencies between the links introduced by this module to improve the overall quality of the produced sub-tree alignments (cf. section 3.3). The only possible explanation for this we can see in the analysis data is that the *rescore* configurations consistently produce fewer links than the non-*rescore* ones (cf. Table 8, Appendix A). These configurations also consistently show a lower recall in the intrinsic evaluation (cf. Table 6) both for lexical and non-lexical links compared to their counterparts. This may be the explanation for the observed results, especially considering the fact that the numbers in Table 8 represent only exact matches to alignments produced by the full-search-based algorithm. I believe that this could be due to the elimination of certain originally nonzero links through the re-scoring process. Whether this is beneficial to the final quality of the parallel treebank or not can only be ascertained through extrinsic evaluation methods, and I will return to this question in section 4.2.

The next largest advantage seen in Tables 8 and 9 is the advantage of using *score2* compared to *score1*. Here you have to consider one important factor, namely that the only difference between the various configurations that I am focusing on—the difference in scoring strategy—is also present for the full-search runs that I use as the benchmark data. Nevertheless, the configurations employing *score2* bar a few exceptions always obtain higher precision, recall and coincidence rates than their *score1* counterparts. This obviously cannot be due to differences on the algorithmic side of the system, but no single factor seems to stand out as the explanation for these results. Theoretically, a change in the scoring strategy can affect which alignment hypotheses obtain a zero score during the initial calculation and it could also lead to having a different set of undecided links and a different

number of undecided links at the end of the alignment process per sentence pair. However, even the data from Table 10 from Appendix A and Table 10 below do not provide obvious explanations for the difference in *score1* and *score2* performance. Therefore, I can only conclude that the *score2* strategy better represents translational equivalencies in general and consequently produces alignment hypotheses that are easier to deal with in the context of the system presented in this work.

Scoring function	all links	lexical links	non-lexical links
score1	28.10	21.95	6.82
score2	28.35	22.13	6.90

Table 10: Average number of initial nonzero alignment hypotheses for the *tree-to-tree* module depending on the scoring function used for the HomeCentre data set

The next performance variation I turn to is the variation between system configurations using the *skip1* module and ones using the *skip2* module. Here a negligible increase in recall values is seen with only a .08 percent points average increase and .39 percent points maximum increase for configuration *tree2tree_21_gs* over *tree2tree_11_gs*. The increase in precision is more palpable with a .45 percent points average increase and .91 percent points maximum increase for the same configurations as before. This improvement in precision is in line with the evaluation against the gold standard and vindicates our prediction that the *skip2* module should lead to better performance. As discussed in section 3.1.2, the *skip1* module allows any link to resolve an existing conflict, while in *skip2* we make sure that the link resolving the conflict will not align a node which is already involved in the conflict. The improvement in performance is also reflected at the slightly higher number of exact matches against the full-search-based algorithm.

Finally, I need to check the effects of the use of the *span1* module. Contrary to my expectations, the configurations using this module fare significantly worse than the ones without it, particularly with regard to preci-

sion. I believe, however, that this is not an evidence for problems with the *span1* module. Rather, the reason for this discrepancy lies in the nature of the *span1* extension. As discussed in section 3.1.2, *span1* introduces a separation in the induction of lexical and non-lexical links. The full-search algorithm, however, derives the maximal link set from a common pool of all nonzero alignment hypotheses which makes direct comparison impossible. I will, therefore, use the extrinsic evaluation results from section 4.2 to give a proper insight into the actual performance of the *span1* module.

In conclusion, the basic trends that can be taken away from this section are that *score2* performs better than *score1*, and that *skip2* leads to better precision than *skip1*. At the same time, the *rescore* module fell short of my expectations and I was unable to properly evaluate the *span1* module. Nevertheless, the greedy-search algorithm comes very close to the best maximal link set. My tests show that in up to 95% of all cases the greedy-search algorithm finds the best maximal link set available for the particular sentence pair. These results are very encouraging and show that the fast greedy-search algorithm produces high-quality results and there is no practical need to use the full-search algorithm, except for comparison and evaluation purposes.

In the next section, I perform extrinsic evaluation of the various configurations of my system and check the extent to which they follow my findings and expectations from the intrinsic evaluation.

4.2. Extrinsic Evaluation

In this section, I use an external system built to use parallel treebanks as a training resource, to evaluate the quality of the treebanks produced by the different configurations of my system. I try to match these results against the intrinsic evaluation presented in section 4.1 and check whether my predictions and expectations are met.

For extrinsic evaluation, I used the Data-Oriented Translation (DOT) system described by Hearne and Way (2006). The system was first trained

using the manually aligned HomeCentre treebank and the output translations were evaluated to acquire baseline scores. I then trained the system on the parallel treebanks automatically generated using my aligner, as well as on the *CMU_TnT* data set and repeated the same evaluation, such that the only difference across runs are the sub-tree alignments.

For testing, I used the six English–French training/test splits for the HomeCentre used by Hearne and Way (2006) for continuity of experimentation. Each test set contains 80 test sentences and each training set contains 730 tree pairs, where the test sentences do not contain unknown words. I evaluated the translation output using three automatic evaluation metrics: BLEU (Papineni et al., 2002), NIST (Doddington, 2002) and METEOR (Banerjee and Lavie, 2005). I averaged the results over the six splits. I also measured test-data coverage of the translation system, i.e. the percentage of test sentences for which full parses were generated on the source side during translation.

I also used the 10,000 sentence pair EuroParl data set described at the beginning of section 4.1. Here I prepared four English–German training/test splits in a manner similar to the one employed for the HomeCentre. Each test set contains 150 test sentences and each training set contains 9,850 tree pairs. The test sentences were restricted to having between 4 and 16 words on the German side, and I also made sure that they did not include words not present in the corresponding training set, so that the results would not be affected by the presence of unknown words.

The BLEU metric (Papineni et al., 2002) is currently one of the most widely used metrics for MT evaluation. This is a string-based metric and it uses a simple calculation of modified precision, i.e. counting the number of *n*-grams[8] from the translation that match the reference. It uses geometric average to combine the scores for *n*-grams of various lengths and it includes

[8] An *n*-gram is a sequence of *n* consecutive tokens in a sentence.

a brevity penalty for translations shorter than the reference. Usually, scores for *n*-grams of length of up to four tokens are calculated. It has to be noted, that the BLEU metric has been developed specifically for use on the document and system level with several reference translations for each translation sentence that needs to be evaluated. Because of this, BLEU has been widely criticised for its inadequate accuracy of evaluation at the sentence level, for example in (Callison-Burch et al., 2006, Owczarzak, 2008).

Developed based on BLEU, the NIST metric (Doddington, 2002) tries to alleviate some of its shortcomings. This is done by factoring in the information score of an *n*-gram, based on its frequency in the document, as less frequent *n*-grams are thought to carry larger part of the meaning of a text and so their correct translation should be awarded higher. Also, NIST uses arithmetic average, rather than geometric, to combine the scores for *n*-grams of up to five tokens in length. Still, NIST is also a document-oriented metric and it also requires multiple reference translations to correlate better to human judgements.

METEOR (Banerjee and Lavie, 2005) differs considerably from BLEU and NIST in that it tries to accommodate both syntactic and lexical differences between the candidate translation and the reference. It operates in several stages, first looking for exact matches and then using stemming to find matches differing only in morphological form and eventually employing WordNet (Fellbaum, 1998) to find matches between synonyms. The final score combines precision and heavily weighted recall with a penalty for non-contiguous matches. METEOR usually shows significantly better correlation with human judgement than BLEU at the segment level. For my experiments, I did not employ the WordNet-based synonymy module of METEOR, as English was the source side for the experiments. The presented scores are only based on exact matches and stemming.

The actual DOT system that I am using is a modification of the one by Hearne and Way (2006). Changes were introduced by the author of this

thesis with the goal of improving the performance of the system and correcting some errors and inconsistencies that I came across during testing. The translation of a sentence with the DOT system consists of four stages that are usually found in tree-based MT systems. First, the sentence is being parsed to obtain an *n*-best list of possible parses. In the case of the DOT system this is done by first performing a *1*-best Viterbi (1967) parse and then generating the remaining *n*-best parses using an algorithm developed by Jiménez and Marzal (2000).

The DOT system includes a fallback mechanism for dealing with sentences that cannot be fully parsed by the *1*-best Viterbi algorithm using the grammar extracted during training. In such cases, an additional root node is added that connects the parse-tree fragments that are generated, to ensure that the transfer process can proceed normally.

The transfer is the second step in the translation process. During transfer the information encoded by the sub-tree links in the parallel treebank is used to find the possible corresponding structures for the available tree fragments on the source side. This process is also guided by the target-side grammar, extracted during training. Finally, once we have the possible target-side derivations that correspond to the source-side parses, we extract the best translations corresponding to the source-side sentence.

It should be noted that the DOT system does not operate with a DOT-style grammar directly. Rather, it uses the Goodman reduction described by Goodman (2003) to construct an equivalent probabilistic context-free grammar (PCFG). For my experiments, I use a grammar-extraction module designed by the author of this thesis that generates the necessary Goodman-style PCFG grammar for use with the DOT system from a given parallel treebank.

I will first look at the results from the HomeCentre evaluation in Tables 1 and 2 in Appendix B and then I will compare them to performance

of the EuroParl data sets, as per Tables 3 and 4 from the same appendix. For clarity, Tables 1 and 2 are colour-coded to show the relevant performance against the gold standard. Namely, scores higher than the gold-standard score have green background, while the ones that are lower have red background. Additionally, Figure 28 presents a graphical comparison between the performance of my system and the CMU and manual alignments on the HomeCentre. Due to the small size of the data sets, statistical significance testing was not performed.

First of all, in Table 1 from Appendix B and on Figure 28 you see that the performance of the automatic systems falls mostly below that of the gold standard on the BLEU and NIST metrics. Still, 26 out of 64 configurations of my system had BLEU scores of 0.52 and higher, which is comparable to the 0.5285 score of the gold standard. The situation with the NIST metric is worse, with only 11 configurations reaching a score of 7.0 and over compared to 7.0632 for the gold standard. Table 2 from Appendix B, however, shows a better picture with over a third of the automatic configurations outperforming the gold standard on the METEOR metric. Also, all configurations of my system produce significantly higher coverage than the gold standard, which should result in more fluent translations.

I believe that the large difference in performance between the BLEU and NIST metrics on one side and the METEOR and coverage metrics on the other, is caused by the inherent differences in the operation of the different metrics. As I discussed earlier, both BLEU and NIST are designed predominantly to perform evaluation on the document level. Also, they perform best when multiple reference translations are available. In my evaluation, however, the small size of the data sets forces me to evaluate on the sentence level, as the test sentences are taken at random from the whole data set. Evaluation on the document level would require the test set to consist of consecutive sentences from a consistent document. The nature of the data sets also means that I only have one reference translation available to

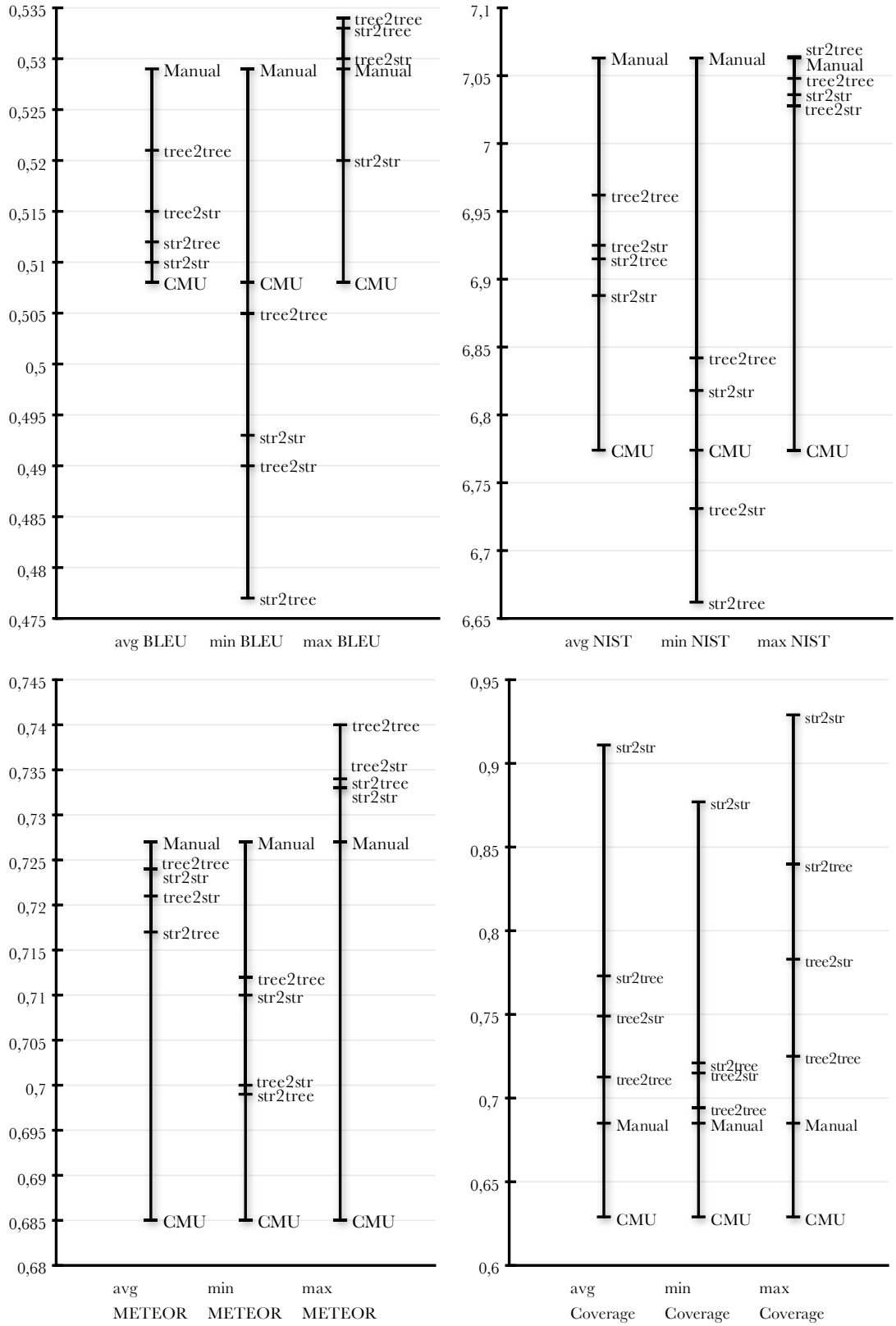


Figure 28: Comparison of the average, minimal and maximal scores achieved for English–to–French translation using various versions of the HomeCentre parallel treebank

work with. I believe that the METEOR metric better represents the actual quality of my system, as it is better suited for operation in such conditions. Finally, the coverage metric does not directly evaluate translation quality. Rather, it is designed specifically to assess the ability of the DOT system to deal with the test data, given the training data it is presented.

Because of these differences in the metrics that I used, I can interpret the evaluation results to mean that the system proposed in this work is a viable alternative to the manual generation of parallel treebanks, due to its similar performance to the gold standard on the BLEU and NIST metrics and better performance on the METEOR metric. This is a very important outcome—especially in view of the significantly improved coverage—given that I am reducing the time needed to prepare data sets that produce translations similar to the gold standard from months to minutes.

The more important comparison that can be taken from here, however, is that the majority of the configurations of my system outperform or perform on par with the CMU system. This is particularly true for the *tree-to-tree* module, which is the one that directly corresponds to the operation of the CMU algorithms. This finding supports my prediction that the *TnT* algorithm from CMU has some inherent flaws that should lead to worse overall performance, for example its over-reliance on the quality of the alignments produced by the word-alignment system used, cf. section 2.2.2. More supporting evidence is found in Figure 28, where the *CMU_TnT* data set leads to the lowest coverage overall; even lower than for the gold standard. It also produces the lowest METEOR score by a significant margin. The low coverage is easily explained by the relatively small number of lexical links produced by this algorithm (see Figure 27) and by the lack of one-to-many links, with respect to the number of tokens covered by the aligned nodes, cf. section 4.1.1. These two factors together result in the lack of a sufficient number of fundamental building blocks needed by the DOT system to generate proper translations.

Another trend seen in Figure 28 is that the *tree-to-tree* configurations have a slight edge over the other modules with respect to BLEU and NIST scores. My interpretation of this fact is that it is beneficial to use parsed data as the basis for a parallel treebank, as it better encodes the syntactic structures that are specific to the language in question.

On the other hand, the data shows that the *tree-to-tree* configurations produce the worst coverage overall (although still significantly better than the gold standard with 2.72 percent points improvement on average), with the *string-to-tree* and *tree-to-string* modules fairing somewhat better (6.03 points improvement on average for the former and 3.63 points on average for the latter), and the *string-to-string* module consistently producing coverage in the region of 90%. I believe that this results from the automatically generated structures being more consistent and better suited for use with DOT overall, due to the larger number of alignments. The important point here is that even if the *string-to-tree*, *tree-to-string* and *string-to-string* modules achieve lower translation scores, they still represent a feasible way of generating parallel treebanks for situations where few resources are available for one or both languages in question.

Now I can turn to the comparative performance of the different configurations of the alignment algorithm using Tables 1 – 4 from Appendix B and Figures 29 – 35. On the *Y* axis of these figures, I mark the difference in the scores achieved using configurations of my system that differ only in the feature that is being investigated. On the *X* axis, I mark the different operation modules of the system, with respect to the type of input data. To mark the actual configurations being tested, I use the convention from Table 5 with a little modification. Namely, a '*' is used as a substitute for the operation module and a '?' substitutes the feature that is being analysed in the abbreviation code. In this way, the label **_21.?_gs* in the first group of values on the top-left graph of Figure 29 marks the value corresponding to the difference in BLEU score between the following two con-

figurations: *tree2tree_21_gs* and *tree2tree_21r_gs*. Because the value is positive, we can conclude that the latter configuration performs better with respect to BLEU than the former, i.e. the inclusion of the *rescore* module has a positive impact in this case. A specific convention used only on Figure 29 is the use of a '.' to signify a configuration where the *span1* module is disabled. That is, **_21.?_gs* corresponds to configurations selecting *skip2* and *score1*, but not *span1*; **_211?_gs*, on the other hand, corresponds to configurations employing *span1* in addition to *skip2* and *score1*.

Looking Figure 29, I first note the lack of a significant drop in performance for the configurations using the *rescore* module that I had expected given the intrinsic evaluation against the full-search-based algorithm (cf. section 4.1.3). In some cases, I even see higher scores that the non-*rescore* counterparts. The explanation that can be given here follows the explanation of the under-performance of the *span1* configurations on the same test, namely that the full-search-based algorithm has no way of representing the interdependence of the produced alignments that is inherent to the *rescore* module. In this case, to optimise the selection process the *rescore* module uses the fact that the greedy-search-based selection module induces alignments in a specific order; for the full-search-based module no such order exists. Therefore, I could not assess the quality of the *rescore* module against the full-search algorithm, but it can be seen here that the ideas behind it are not without merit. Still, in its present form the use of this module leads to a significant drop-off in coverage which is probably one of the main reasons for the low translation scores. Thus, the use of this module cannot currently be justified.

Looking at Figures 30 and 31, it can be seen that—with very few exceptions, notably for the *string-to-string* case—configurations employing the *score1* scoring strategy lead to lower translation scores across all metrics compared to their *score2* counterparts. The trend is sustained for the HomeCentre

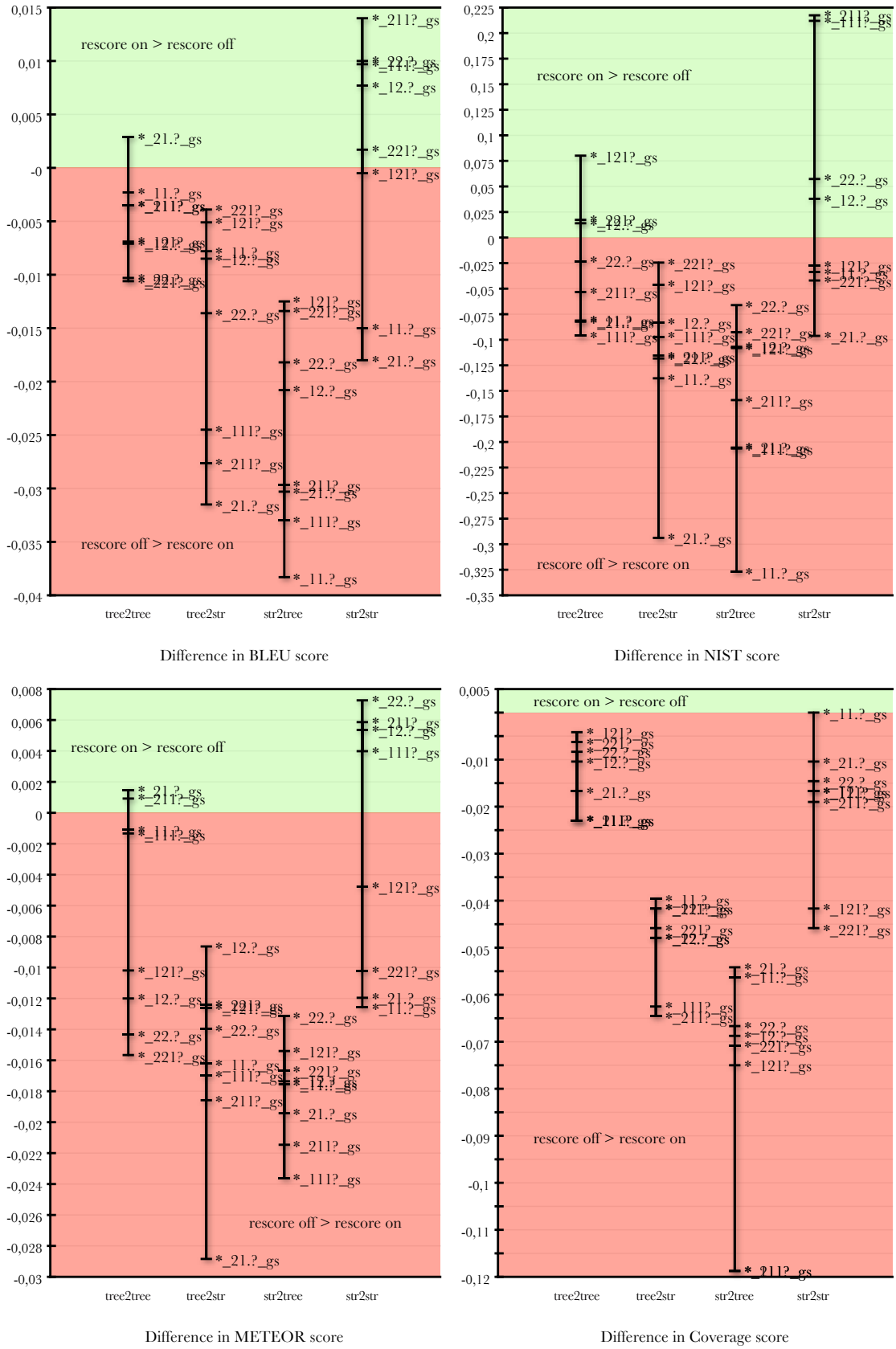


Figure 29: Overview of the impact of the use of *rescore* in English–to–French translation using various versions of the HomeCentre parallel treebank

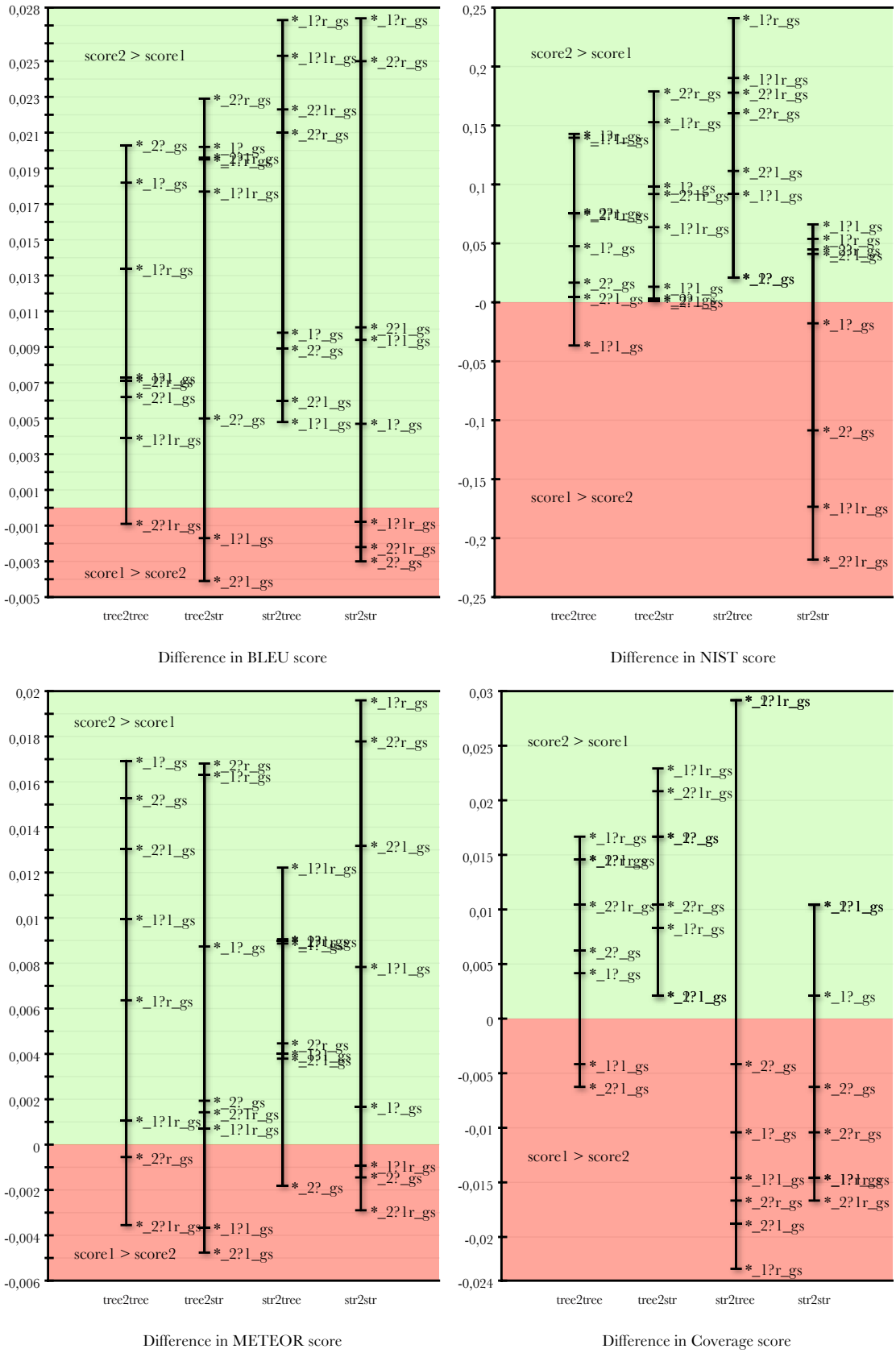


Figure 30: Overview of the impact of the use of *score2* over *score1* in English–to–French translation using various versions of the HomeCentre parallel treebank

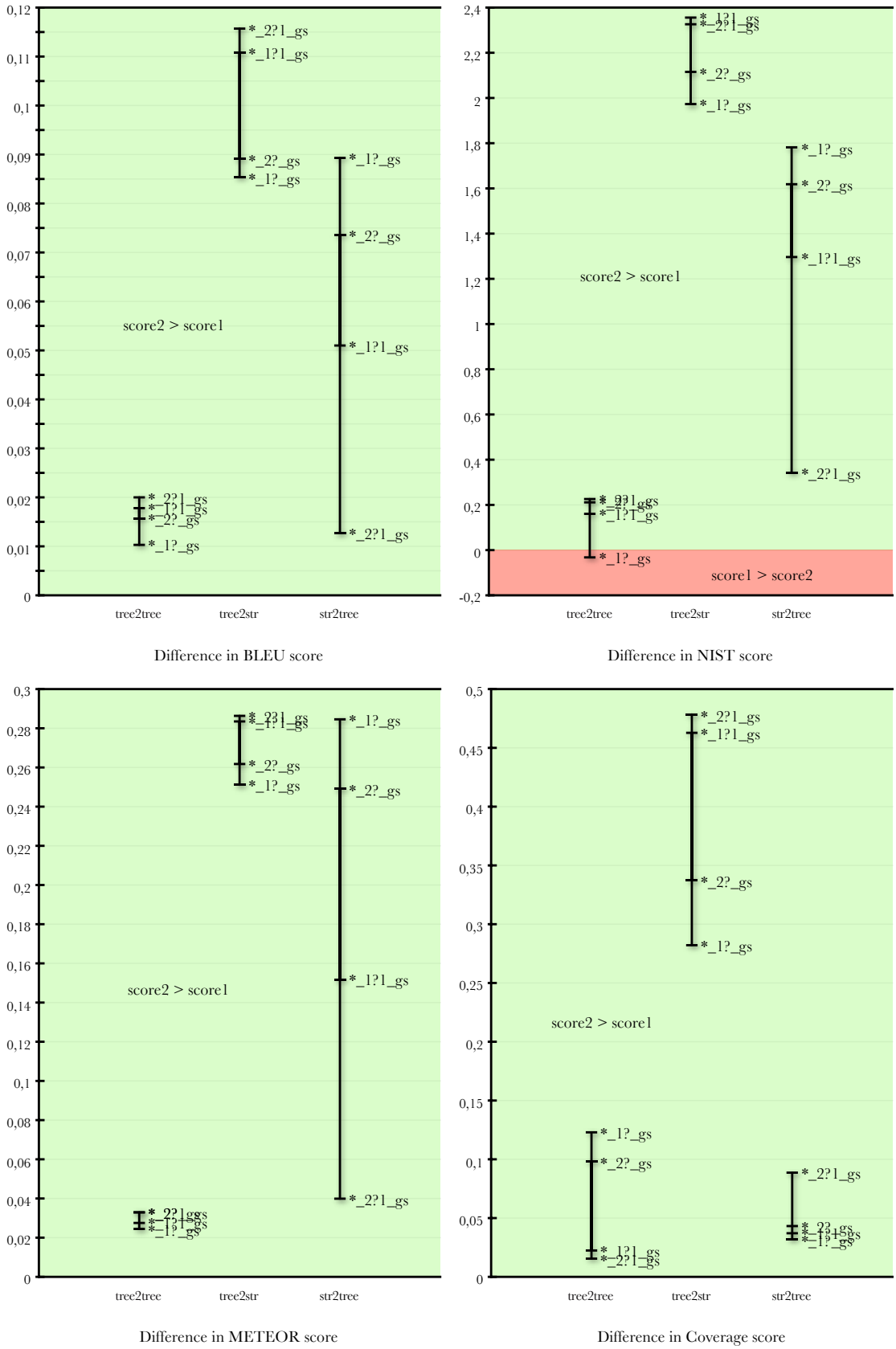


Figure 31: Overview of the impact of the use of *score2* over *score1* in English–to–German translation using various versions of the EuroParl parallel treebank

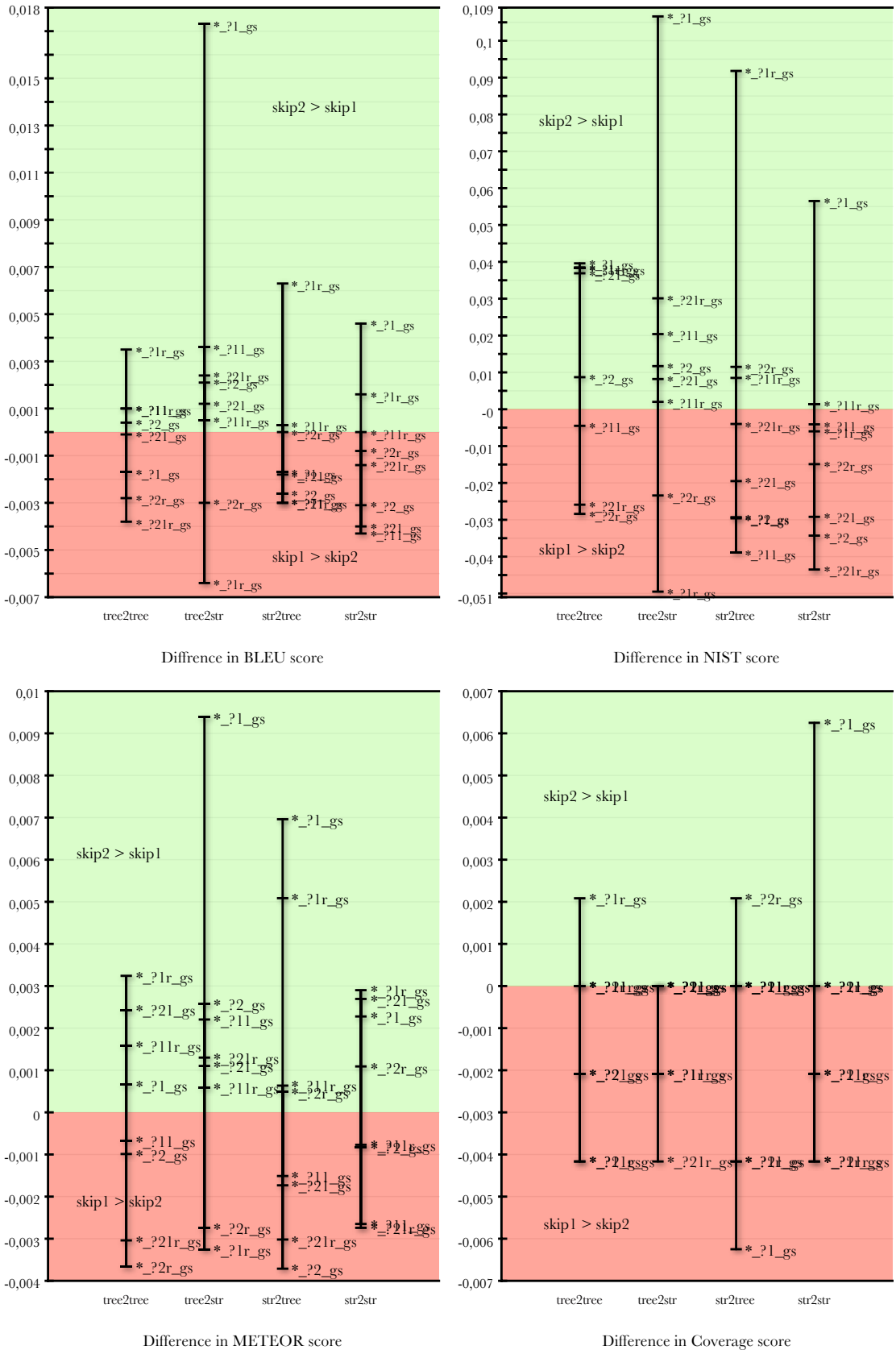


Figure 32: Overview of the impact of the use of *skip2* over *skip1* in English–to–French translation using various versions of the HomeCentre parallel treebank

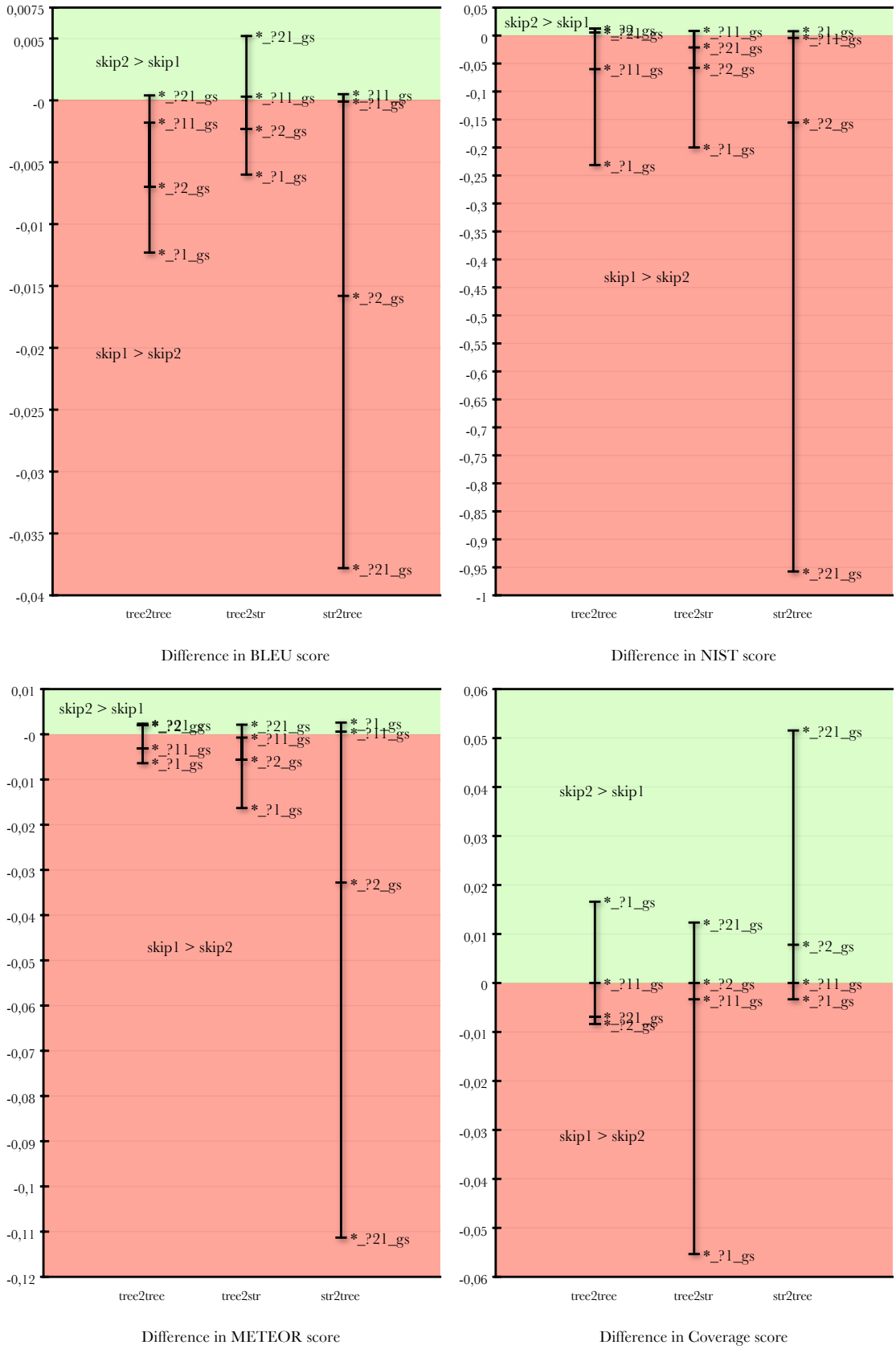


Figure 33: Overview of the impact of the use of *skip2* over *skip1* in English–to–German translation using various versions of the EuroParl parallel treebank

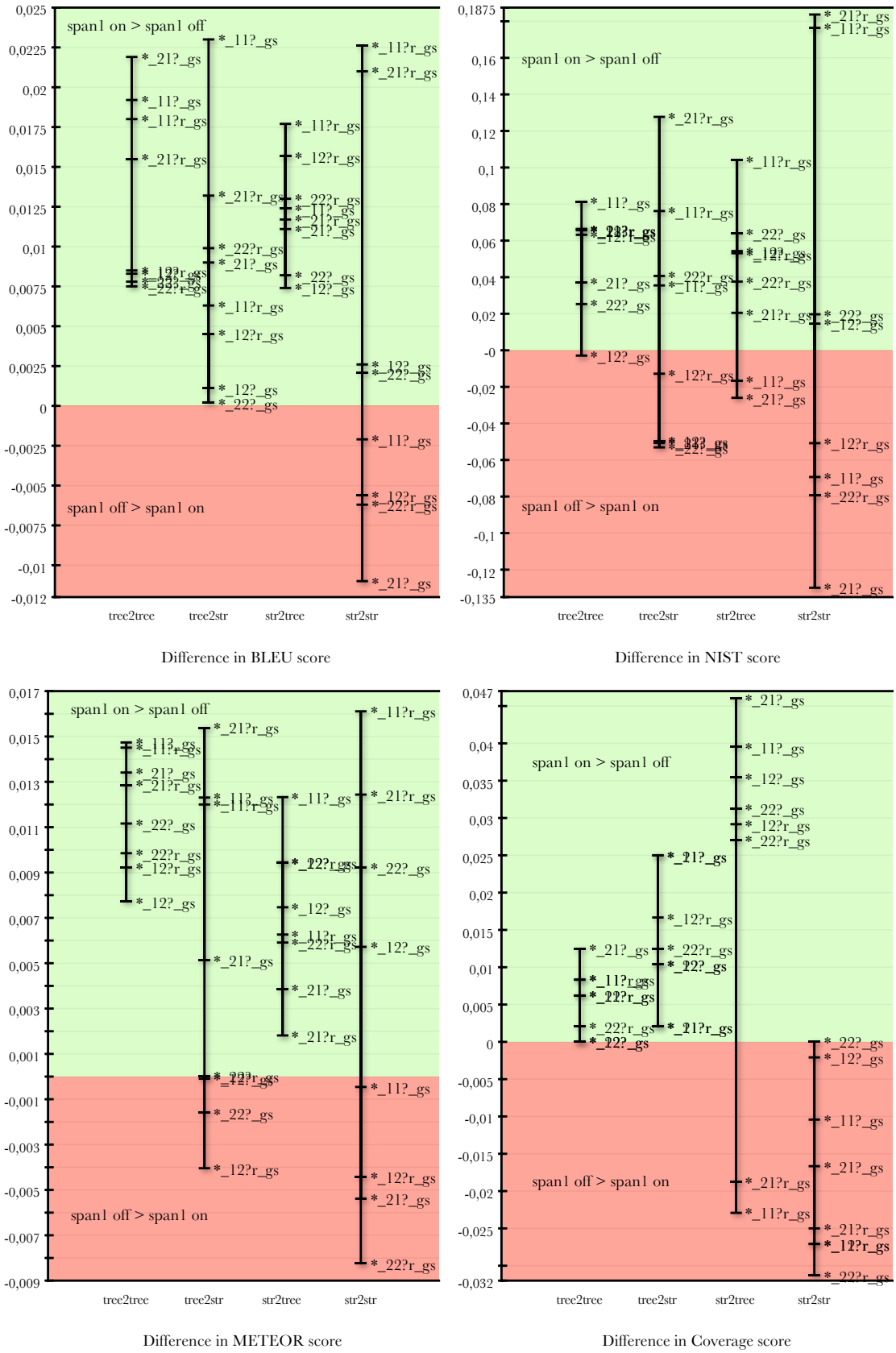


Figure 34: Overview of the impact of the use of *span1* in English–to–French translation using various versions of the HomeCentre parallel treebank

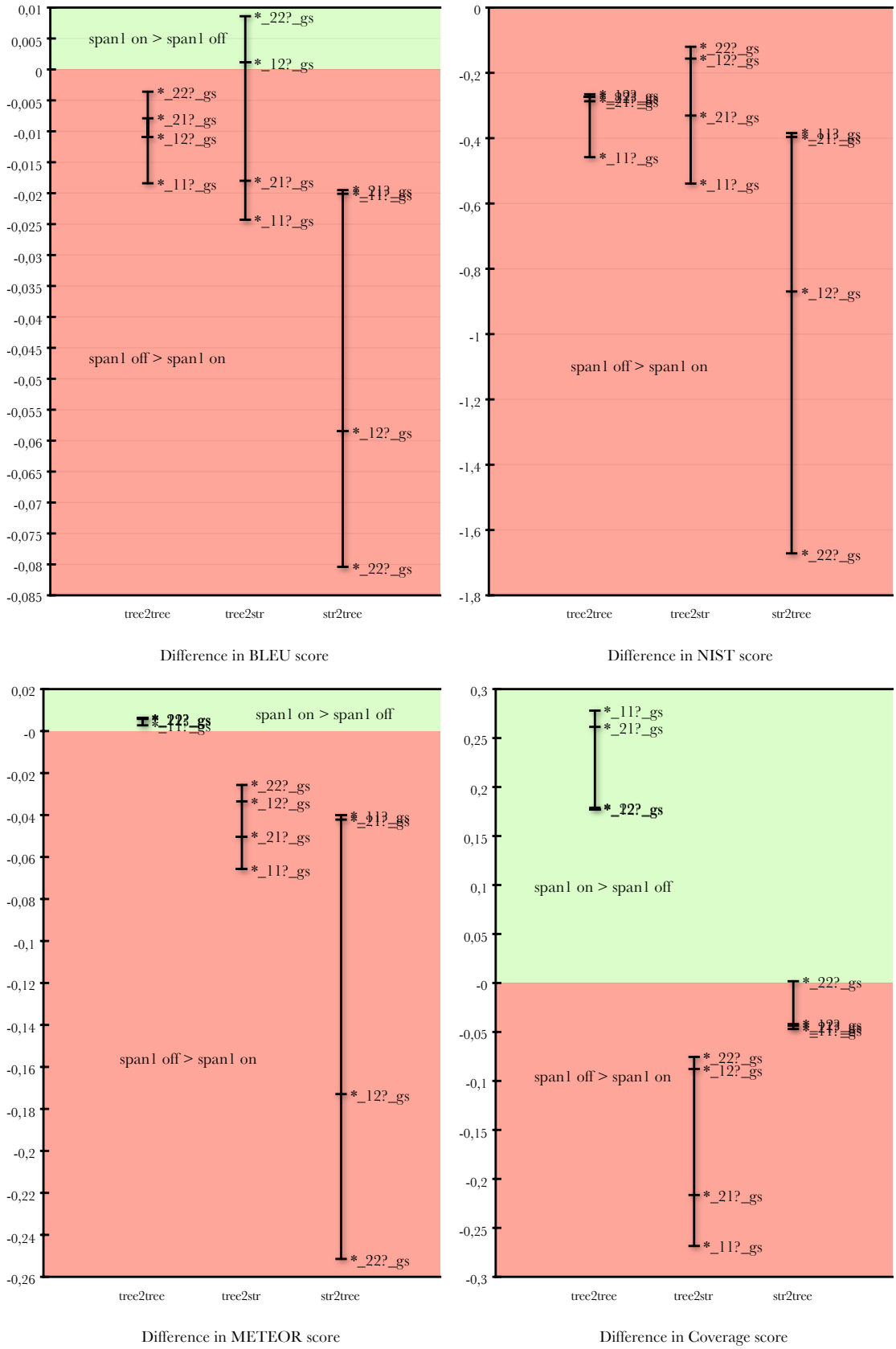


Figure 35: Overview of the impact of the use of *span1* in English–to–German translation using various versions of the EuroParl parallel treebank

and is especially prominent for the EuroParl data set. This follows my findings from the intrinsic evaluation in section 4.1 and reinforces my conclusions that *score2* is the better scoring strategy that should be employed for the generation of parallel treebanks.

As detailed in section 4.1.3, I expected to see at least a slight advantage for configurations using *skip2* over their counterparts using *skip1*. However, the data on Figures 32 and 33 are inconclusive with a lot of variation, even when I only consider configurations with *score2*. The *skip2* module seems to perform particularly poorly on the EuroParl data set, however the differences in the scores remain very narrow. Therefore, I can only recommend the use of *skip2* over *skip1* on theoretical grounds and following the intrinsic evaluation.

For the *span1* module, I see significantly better results for *tree-to-tree* operation for the HomeCentre data set on Figure 34. However, the results are mostly inconclusive for the other translation modules for HomeCentre and strongly negative for EuroParl, as seen on Figure 35. The trend for the *tree-to-tree* use of the *span1* module follows my findings from the intrinsic evaluation against the gold standard, so I have to look elsewhere for an explanation of the discrepancy with the performance of the *tree-to-string*, *string-to-tree* and *string-to-string* modules. A possible clue can be found in Tables 10 and 11 in Appendix A. Here I see a general trend for configurations using *span1* to produce a higher percentage of non-lexical links on average. Although this trend can also be found for the *tree-to-tree* module, I believe that there it brings the non-lexical/lexical links distribution closer to the ideal for the data set, whereas when operating in any of the other translation modules the alignments are produced with an unnecessarily strong bias towards non-lexical links. In my opinion, this is due to the fact that the *tree-to-string*, *string-to-tree* and especially the *string-to-string* modules do not in advance have the structure that is necessary to guide the operation of the *span1* module.

Therefore, I can conclude that the *span1* module should only be used when operating in *tree-to-tree* mode, where it is in general beneficial to the quality of the resulting parallel treebank and that its use should be avoided in all other cases, as they do not provide the input necessary for its proper operation.

To conclude, most of my expectations for the performance of the automatic sub-tree aligners were met, and for the ones that were not I found explanations in the data that I have at my disposal. Most importantly, I saw that my system performs similarly or better than the hand-crafted gold standard on several metrics, with significantly better coverage, making it a real alternative to the manual generation of parallel treebanks.

4.3. Conclusions

The results obtained from the intrinsic and extrinsic evaluations show that the methods described in this chapter produce high-quality parallel treebanks. Using the automatically generated treebanks, a DOT system produces results with similar translation quality and better coverage compared to its performance using manually aligned data. This makes my methods a good alternative to the manual construction of parallel treebanks, especially considering that they require only a fraction of the effort and time that would otherwise be needed to manually prepare the same amount of data.

Following my evaluation, I can recommend the use of the *221_gs* configuration with the *tree-to-tree* module and the *22_gs* configuration with the *tree-to-string*, *string-to-tree* and *string-to-string* modules. The use of the full-search-based algorithm for real-life tasks is strongly discouraged due to its time requirements and due to the fact that comparable results can be achieved using the greedy-search-based selection. The *skip2* module and the *score2* scoring function are always preferable to *skip1* and *score1*, while the *span1* module is only useful for *tree-to-tree* operation. The use the *rescore* mod-

ule is currently discouraged, although a new module designed along similar lines could be developed in the future that would lead to better results.

It was also seen that it is possible to generate high-quality parallel treebanks even when some or all the data cannot be parsed in advance. This should allow the application of rich syntactic models of translation even to under-resourced languages without adding a significant pre-processing overhead.

Next, I will propose a few extensions to the system that might improve its performance, as well as a few additional evaluation strategies that might be used to better understand and assess the performance of my sub-tree aligner.

5. Further Avenues for Development and Evaluation

In the previous chapter, I evaluated my system for the automatic generation of parallel treebanks and concluded that it is a viable and readily available alternative to the manual annotation of parallel treebanks, even for under-resourced languages.

Still, my system cannot be claimed to be perfect and here I look at several possible extensions and improvements that, while not necessary, may be developed in the future. I also propose some additional evaluation and analysis experiments that might help with the assessment of the qualities of the generated parallel treebanks.

5.1. Improvements to the algorithms

I start with a few suggestions about possible improvements to the underlying algorithm of my system. As was discussed in section 3.1.2, there are cases in which my greedy-search-based algorithm cannot disambiguate all available alignment hypotheses and has to leave some unrealised. These unrealised hypotheses, however, are currently not available to the user of the system. One possible way of giving access to this information would be to output of all possible ambiguous link combinations, thus allowing the user to deal with the ambiguity within the application that uses the generated parallel treebanks. Another option would be to randomly select an alignment from each available tie and presenting an unambiguous sentence pair to the user. Such a process might not necessarily produce the best alignments, but it will increase the number of alignments per sentence pair, which could lead to better translation performance.

The system presented in this thesis is designed to use a probabilistic bilingual dictionary as its source for word-level translational equivalence data. So far I have suggested that an automatic word alignment tool like

GIZA++ (Och and Ney, 2003) be used to obtain these data and this is what I myself employed for the experiments presented in Chapter 4. However, such automatic tools do make errors and these errors can then affect the quality of the parallel treebanks generated using my system. One such very common error I have observed is the alignment of content words to punctuation. For example, aligning *door* to *!* can never be a proper alignment, but a word-alignment tool might produce a nonzero equivalence score for it. This issue can be mitigated, for example, by compiling two lists containing the appropriate punctuation for the languages that are being aligned and modifying the alignment algorithm so that it filters the available word-alignment data according to these lists. Another option that does not require the modification of the software presented here is to pre-process the word-alignment data by filtering it according to the punctuation lists and redistributing the probability mass of the excluded word alignments.

The system presented in this thesis was developed to operate specifically on phrase-structure trees. However, considerable amount of contemporary research in MT is moving towards the use of dependency structures instead, as they are considered to exhibit less variation across languages. Because of this, there has been interest in using the system I developed to generate parallel treebanks with dependency-structure analysis. Currently, this can be done by converting the dependency structures into phrase-structure trees. A good example of this is presented by Hearne et al. (2008).

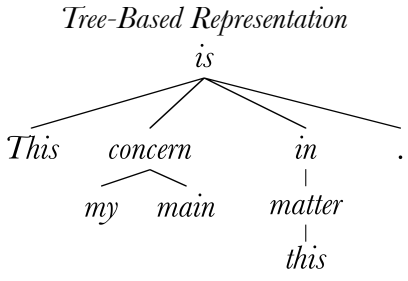


Figure 36: An example of dependency-based analysis of a sentence

However, the conversion method used by Hearne et al. does not allow the encoding of non-projective dependency structures. Also, such converted structures do not necessarily contain the alignments that would be present between the original dependency structures. As was seen in Figure 2 in the beginning of Chapter 2 (the tree-based representation from which is repeated in Figure 36 for convenience), the nodes of a dependency tree correspond to tokens from the sentence that is being analysed. In the converted structure employed by Hearne et al. (2008), however, the non-terminal nodes from the tree in Figure 36 (namely *is*, *concern* , *in* and *matter*) occur both as non-terminal and as terminal nodes, as can be seen in Figure 37.

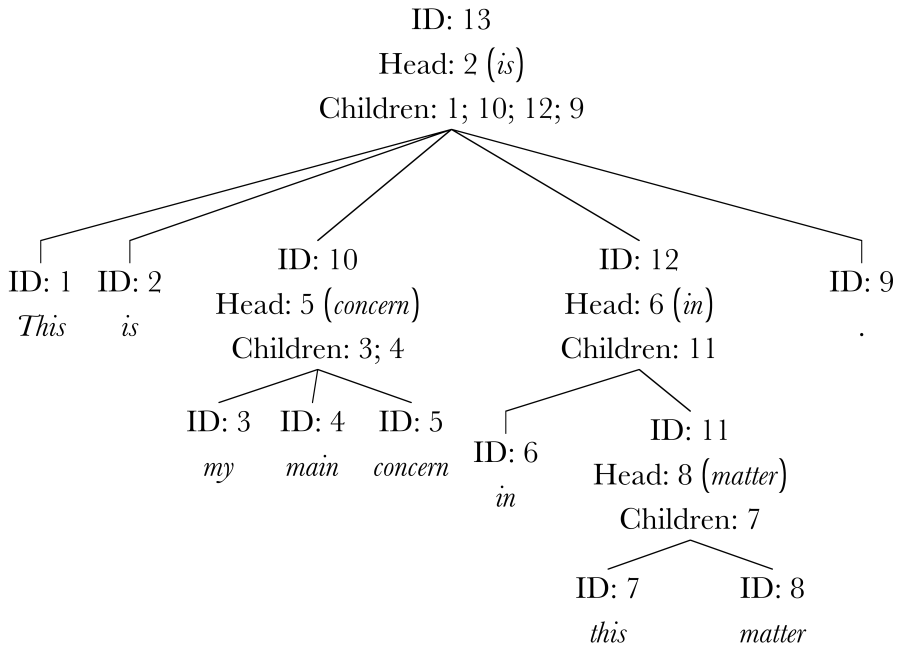


Figure 37: A phrase-structure-based conversion of the dependency structure from Figure 36, adapted from (Hearne et al., 2008)

This means that my sub-tree aligner in its present form might produce alignments both for the non-terminal and terminal versions of these tokens. While such results might be regarded as beneficial for the extraction of phrase pairs—which is the main goal of Hearne et al. (2008)—the generated parallel treebank cannot be converted back to dependency structures without deciding how to deal with such multiple alignments. To mitigate problems like this, the sub-tree aligner should be able to operate directly on

dependency structures without any conversion. My system has a modular design, where there is a separate module responsible for the storage of phrase-based tree structures and for the retrieval of their properties. Because of this, a new module that operates on dependency structures can be developed that would replace the module operating on phrase-based trees. The rest of the implementation can be kept intact and would be able to perform alignment of dependency structures directly and so generate proper dependency-structure-based parallel treebanks.

5.2. Further Analysis and Evaluation

Now I will look at some further experiments that could help better understand the operation of my sub-tree alignment system.

As mentioned in the previous section, currently I only employ word-alignment data derived from GIZA++. These data, however, may be noisy and contain erroneous alignments. Because the quality of the parallel treebanks generated using my system depends directly on the quality of the available word-alignment data, it would be interesting to perform evaluation experiments using other probabilistic bilingual dictionaries and check their impact on the performance of the generated parallel treebanks. These might either be derived using automatic word-alignment tools different from GIZA++, or could be specifically produced high-quality electronic bilingual dictionaries.

Currently Inversion-Transduction Grammar (ITG – Wu, 2000) is a popular architecture for the extraction of word- and phrase-alignment data from parallel texts. As discussed in section 3.2, the *string-to-string* module of my system is closely related to ITG in its operation. Because of this, it would be interesting to explore their relation in greater detail. Most important would be to investigate the level of isomorphism present in the structures generated by my *string-to-string* algorithm and relate that to the structures generated by ITG. If possible, this evaluation should be performed on

a data set for which an ITG parse can be generated. This experiment should also include an analysis of the properties of the different structures, like depth of the generated trees, average number of nodes, etc.

Another possible evaluation would be to estimate the extent to which the parallel treebanks generated using my system properly encode the translational equivalencies between languages, especially for the *string-to-tree*, *tree-to-string* and *string-to-string* modules, where the aligner itself generates structure. This evaluation can be performed using analysis following (Hearne et al., 2007), that is investigating the ability of the sub-tree aligner to handle hard cases of translational divergence like head switching, nominalisation, relation changing, etc. The evaluation should, however, also include an analysis of the impact of parser quality on the quality of the generated parallel treebanks.

In section 4.1.2 was observed that, while there is a significant overlap, my system and the *TnT* module by Ambati and Lavie (2008) produce differing sets of alignments. It would be useful to investigate this in greater detail, especially on larger data sets. Such an analysis might suggest possible ways of combining the alignments produced by the two systems in order to achieve better performance at syntax- and phrase-based machine translation tasks.

Finally, in section 4.1.3 was seen that the full-search-based algorithm cannot be used to properly benchmark the quality of the *span1* module. Therefore, I want to suggest that functionality equivalent to the operation of the *span1* module be implemented for the full-search-based selection algorithm. However, this is not a crucial development, as evaluation using the full-search algorithm is only possible for small data sets and with limited sentence length. The implementation of this functionality will simply complete the benchmark tools that were used in this thesis.

The improvements suggested in section 5.2 can make the system presented in this thesis an even better and more useful tool to the MT commu-

nity. Given that the system was released as an open-source project at the Open-source Convention at the Third MT Marathon in Prague, the Czech Republic (Zhechev, 2009), I believe that these improvements—as well as many more that I have not thought about—will be contributed by the wider computational linguistics community.

The implementation of the experiments suggested in this section, on the other hand, will allow for the assessment of the qualities of the system presented here from several new perspectives. This could convince a larger number of people of the usefulness of a tool like this and might help accelerate its adoption by MT researchers.

6. Conclusion

In this work, I presented an open-source platform for the automatic generation of parallel treebanks from parallel corpora. I discussed algorithms both for cases in which monolingual phrase-structure parsers exist for both languages and for cases in which such parsers are not available. Through both intrinsic and extrinsic evaluations I showed that the parallel treebanks created with the methods described in this work are of a high quality and can successfully be used as training data for data-oriented machine translation systems. The parallel treebanks can also be used for other statistical MT applications and for translation studies.

First in Chapter 2, I conducted a survey of existing methods for subsentential alignment and hand-crafted parallel treebanks. The outcome of this survey is that among the existing parallel treebanks that can be used for MT tasks (Čmejrek et al., 2004, Uchimoto et al., 2004, Samuelsson and Volk, 2006), none are of sufficient size for use in practice. Also, given the difficulty of the task of manually annotating parallel treebanks, I do not expect a significant increase in the size of the hand-crafted parallel treebanks in the foreseeable future.

Among the reviewed sub-tree alignment systems, none were designed purposefully for the generation of parallel treebanks. Still, a number of them can (to a certain extent) be converted for this task (Kaji et al., 1992, Eisner, 2003, Ambati and Lavie, 2008). However, the system of Eisner is computationally complex due to the use of the EM algorithm and the systems of Kaji et al. and Ambati and Lavie suffer from over-reliance on the correctness of the word-alignment data that they use to induce subsentential alignments. This, combined with the scarcity of hand-crafted parallel treebanks, led to the conclusion that a new tool needs to be developed that would enable the generation of parallel treebanks that supersede any such existing hand-crafted treebanks both in terms of size and quality.

In Chapter 3, I described the underlying design of the system that I developed to tackle the problem stated above. I presented the different configurations that the system can use (section 3.1), as well as the four main operation modes: *tree-to-tree*, *tree-to-string*, *string-to-tree* and *string-to-string* (section 3.2). Both a full-search-based algorithm and a fast, robust greedy-search-based alternative for the determination of the best set of alignments per sentence pair were developed and are described in this chapter (section 3.1). Considering the time and space complexity of the algorithms, analysed in section 3.4, I regard the system presented in this work to be a useful tool able to generate parallel treebanks from very large parallel corpora. Although my system can operate in completely unsupervised manner using only the output of open-source tools (eg. GIZA++ – Och and Ney, 2003), it can benefit greatly from the pre-processing of the data with monolingual parsers and/or POS taggers. Overall, the system I presented is theoretically sound and has good potential.

In Chapter 4, I performed extensive evaluation and analysis of the parallel treebanks produced using my system for two different data sets, the HomeCentre and an excerpt from EuroParl. The results that I obtained show that the automatically generated parallel treebanks are of a high quality and can be used effectively as training data for syntax-based MT systems. A data-oriented translation system (Hearne and Way, 2006) trained on these parallel treebanks achieves similar translation quality and significantly better coverage compared to its performance using manually-aligned data.

Following my evaluation, I established that the *skip2* module and the *score2* scoring function are always preferable to *skip1* and *score1*, while the *span1* module is only useful for *tree-to-tree* operation and the usefulness of the *rescore* module has to be evaluated for the particular data set which is being aligned (see Chapter 3 for details on the algorithms used). I found that my methods are a good alternative to the manual construction of parallel tree-

banks, especially considering that they require only a fraction of the time and effort that would otherwise be expended. The ability of my system to operate in an unsupervised manner from parallel corpora allows the application of rich syntactic models of translation even to under-resourced languages. Therefore, I regard the platform presented in this thesis to be an effective tool for further research in the field of syntax-augmented MT.

Finally, I suggested some possible improvements to the system in Chapter 5. I expect some or all of these additions to be contributed by the open-source computational linguistics community and to make the system an even better and more useful tool. I also discussed some additional evaluation and analysis experiments that can add several new perspectives to the assessment of the properties of the generated parallel treebanks. This could improve understanding of the methods that I employ and make my tool more attractive to the machine translation community.

My system has already been made publicly available and can be accessed at `http://ventsislavzhechev.eu/Home/Software/Software.html`
I hope that it will receive widespread adoption and that it will be a useful tool to many researchers.

Bibliography

Ahrenberg, Lars. 2007. LinES: An English–Swedish Parallel Treebank. In *Proceedings of the 16th Nordic Conference of Computational Linguistics (NODALIDA '07)*, eds. Joakim Nivre, Heiki-Jaan Kaalep, Kadri Muischnek and Mare Koit, pp. 270–274. Tartu, Estonia.

Ambati, Vamshi and Alon Lavie. 2008. Improving Syntax Driven Translation Models by Re-structuring Divergent and Non-isomorphic Parse Tree Structures. In *Proceedings of the Student Research Workshop at the Eighth Conference of the Association for Machine Translation in the Americas (AMTA '08)*. Waikiki, HI.

Baker, James K. 1979. Trainable Grammars for Speech Recognition. In *Speech Communication Papers Presented at the 97th Meeting of the Acoustical Society of America*, eds. Jared J. Wolf and Dennis H. Klatt, pp. 547–550. Cambridge, MA: MIT.

Banerjee, Satanjeev and Alon Lavie. 2005. METEOR: An Automatic Metric for MT Evaluation with Improved Correlation with Human Judgements. In *Proceedings of the Workshop on Intrinsic and Extrinsic Evaluation Measures for MT and/or Summarization at the 43rd Annual Meeting of the Association for Computational Linguistics (ACL '05)*, pp. 65–72. Ann Arbor, MI.

Bellow, Saul. 1976. To Jerusalem and Back: A Personal Account. New York: Viking Press.

Bikel, Daniel M. 2002. Design of a Multi-Lingual, Parallel-Processing Statistical Parsing Engine. In *Proceedings of the Third Annual Human Lnaugage Technology Conference (HLT '02)*, pp. 24–28. San Diego, CA.

Blum-Kulka, Shoshana. 1986. Shifts of cohesion and coherence in Translation. In *Interlingual and Intercultural Communication* , eds. Juliane Hause and Shoshana Blum-Kulka, pp. 17–35. Tübingen, Germany: Gunter Narr.

Brants, Sabine, Stefanie Dipper, Silvia Hansen, Wolfgang Lezius and George Smith. 2002. The TIGER Treebank. In *Proceedings of the First Workshop on Treebanks and Linguistic Theories (TLT '02)*, eds. Erhard Hinrichs and Kiril Simov, pp. 24–41. Sozopol, Bulgaria.

Bresnan, Joan. 2001. Lexical-Functional Syntax: *Blackwell Textbooks in Linguistics*. Oxford: Blackwell.

Callison-Burch, Chris, Miles Osborne and Philipp Koehn. 2006. Re-evaluating the Role of BLEU in Machine Translation Research. In *Proceedings of the 11th Conference of the European Chapter of the Association of Computational Linguistics (EACL '06)*, pp. 249–256. Oslo, Norway.

Chiang, David. 2007. Hierarchical Phrase-Based Translation. *Computational Linguistics*, 33 (2): 201–228.

Čmejrek, Martin, Jan Cuřín, Jiří Havelka, Jan Hajič and Vladislav Kuboň. 2004. Prague Czech–English Dependency Treebank: Syntactically Annotated Resources for Machine Translation. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC '04)*. Lisbon, Portugal.

Cocke, John and Jacob T. Schwartz. 1970. Programming Languages and their Compilers: Preliminary Notes. New York, NY: Courant Institute of Mathematical Sciences, New York University.

Cyrus, Lea. 2006. Building a resource for studying translation shifts. In *Proceedings of the 5th Conference of Language Resources and Evaluation (LREC '06)*, pp. 1240–1245. Genoa, Italy.

Dalrymple, Mary E. 2001. Lexical-Functional Grammar: *Syntax and Semantics*, vol. 34. New York, NY: Academic Press.

Dempster, Arthur P., Nan M. Laird and Donald B. Rubin. 1977. Maximum Likelihood from Incomplete Data via the *EM* Algorithm. *Journal of the Royal Statistical Society, Series B (Methodological)*, 39 (1): 1–38.

Ding, Yuan, Dan Gildea and Martha Palmer. 2003. An Algorithm for Word-Level Alignment of Parallel Dependency Trees. In *Proceedings of MT Summit IX*, pp. 95–101. New Orleans, LA.

Doddington, George. 2002. Automatic Evaluation of Machine Translation Quality Using N-Gram Co-Occurrence Statistics. In *Proceedings of the ARPA Workshop on Human Language Technology*, pp. 128–132. San Diego, CA.

Dras, Mark and Chung-hye Han. 2002. Korean–English MT and S-TAG. In *Proceedings of the Sixth International Workshop on Tree Adjoining Grammars and Related Formalisms (TAG+ '02)*, pp. 206–219. Venice, Italy.

Eisner, Jason. 2003. Learning Non-Isomorphic Tree Mappings for Machine Translation. In *41st Annual Meeting of the Association for Computational Linguistics (ACL '03), Companion Volume*, pp. 205–208. Sapporo, Japan.

Fellbaum, Christiane ed. 1998. WordNet. An Electronic Lexical Database. Cambridge, MA: MIT Press.

Gaarder, Jostein. 1991. Sofies verden: Roman om filosofiens historie. Oslo: Aschehoug.

Gildea, Daniel. 2003. Loosely Tree-Based Alignment for Machine Translation. In *41st Annual Meeting of the Association of Computational Linguistics (ACL '03)*, pp. 80–87. Sapporo, Japan.

Goodman, Joshua T. 2003. Efficient Parsing of DOP with PCFG-Reductions. In *Data-Oriented Parsing*, eds. Rens Bod, Remko Scha and Khalil Sima'an, pp. 125–146. Stanford, CA: CSLI Publications.

Grishman, Ralph. 1994. Iterative Alignment of Syntactic Structures for a Bilingual Corpus. In *Proceedings of the Second Annual Workshop on Very Large Corpora (WVLC '94)*, pp. 57–68. Kyoto, Japan.

Groves, Declan, Mary Hearne and Andy Way. 2004. Robust Sub-Sentential Alignment of Phrase-Structure Trees. In *Proceedings of the 20th International Conference on Computational Linguistics (CoLing '04)*, pp. 1072–1078. Geneva, Switzerland.

Hajičová, Eva. 2000. Dependency-Based Underlying-Structure Tagging of a Very Large Czech Corpus. *TAL (Special Issue Grammaires de Dépendance / Dependency Grammars)*, 41 (1): 47–66.

Han, Chung-hye, Na-Rae Han and Eon-Suk Ko. 2001. Bracketing Guidelines for Penn Korean Treebank: University of Pennsylvania Institute for Research in Cognitive Science. Report No. IRCS-01-10.

Han, Chung-hye, Na-Rae Han, Eon-Suk Ko and Martha Palmer. 2002. Development and Evaluation of a Korean Treebank and its Application to NLP. In *Proceedings of the 3rd International Conference on Language Resources and Evaluation (LREC '02)*, pp. 1635–1642. Las Palmas, Canary Islands, Spain.

Hansen-Schirra, Silvia, Stella Neumann and Mihaela Vela. 2006. Multi-dimensional Annotation and Alignment in an English–German Translation Corpus. In *Proceedings of the workshop on Multi-dimensional Markup in Natural Language Processing (NLPXML '06)*, pp. 35–42. Trento, Italy.

Hearne, Mary. 2005. Data-Oriented Models of Parsing and Translation. School of Computing, Dublin City University: PhD Thesis. Dublin, Ireland.

Hearne, Mary and Andy Way. 2006. Disambiguation Strategies for Data-Oriented Translation. In *Proceedings of the 11th Conference of the European Association for Machine Translation (EAMT '06)*, pp. 59–68. Oslo, Norway.

Hearne, Mary, John Tinsley, Ventsislav Zhechev and Andy Way. 2007. Capturing Translational Divergences with a Statistical Tree-to-Tree Aligner. In *Proceedings of the 11th International Conference on Theoretical and Methodological Issues in Machine Translation (TMI '07)*, eds. Andy Way and Barbara Gawronska. vol. 2007:1, pp. 85–94. Skövde, Sweden: Skövde University Studies in Informatics.

Hearne, Mary, Sylwia Ozdowska and John Tinsley. 2008. Comparing Constituency and Dependency Representations for SMT Phrase-Extraction. In *Actes de la 15éme Conférence Annuelle sur le Traitement Automatique des Langues Naturelles (TALN '08)*. Avignon, France.

Imamura, Kenji. 2001. Hierarchical Phrase Alignment Harmonized with Parsing. In *Proceedings of the Sixth Natural Language Processing Pacific Rim Symposium (NLPRS '01)*, pp. 377–384. Tokyo, Japan.

Jiménez, Víctor M. and Andrés Marzal. 2000. Computation of the *N* Best Parse Trees for Weighted and Stochastic Context-Free Grammars. In *Proceedings of the Joint IAPR International Workshops on Advances in Pattern Recognition*, pp. 183–192. London, UK: Springer.

Kaji, Hiroyuki, Yuuko Kida and Yasutsugu Morimoto. 1992. Learning Translation Templates from Bilingual Text. In *Proceedings of the 15th [sic] International Conference on Computational Linguistics (CoLing '92)*, ed. Christian Boitet. vol. 2, pp. 672–678. Nantes, France.

Kaplan, Ronald M. and Joan Bresnan. 1982. Lexical-Functional Grammar: A Formal System for Grammatical Representation. In *The Mental Representation of Grammatical Relations*, ed. Joan Bresnan, pp. 173–281. Cambridge, MA: The MIT Press.

Kasami, Tadao. 1965. An Efficient Recognition and Syntax-Analysis Algorithm for Context-Free Languages. In *Scientific Report AFCRL-65-758*. Bedford, MA: Air Force Cambridge Research Lab.

Koehn, Philipp, Franz Josef Och and Daniel Marcu. 2003. Statistical Phrase-Based Translation. In *Proceedings of HLT-NAACL '03*, pp. 48–54. Edmonton, AL, Canada.

Koehn, Philipp. 2005. Europarl: A Parallel Corpus for Statistical Machine Translation. In *Proceedings of the Machine Translation Summit X*, pp. 79–86. Phuket, Thailand.

Koehn, Philipp, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin and Evan Herbst. 2007. Moses: Open Source Toolkit for Statistical Machine Translation. In *Proceedings of the Demo and Poster Sessions of the 45th Annual Meeting of the Association for Computational Linguistics (ACL '07)*, pp. 177–180. Prague, Czech Republic.

Lavie, Alon. 2008. Stat-XFER: A General Search-based Syntax-driven Framework for Machine Translation. In *Proceedings of the 9th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing '08)*, ed. Alexander F. Gelbukh, pp. 362–375. Vol. 4919/2008 of *Lecture Notes in Computer Science*. Haifa, Israel: Springer.

Li, Zhifei, Chris Callison-Burch, Sanjeev Khudanpur and Wren Thornton. 2009. Decoding in Joshua: Open Source, Parsing-Based Machine Translation. *The Prague Bulletin of Mathematical Linguistics*, 91: 47–56.

Lü, Yajuan, Ming Zhou, Sheng Li, Chang-Ning Huang and Tiejun Zhao. 2001. Automatic Translation Template Acquisition Based on Bilingual Structure Alignment. In *Computational Linguistics and Chinese Language Processing*. vol. 6, No.1, pp. 83–108. China.

Marcus, Mitchell, Beatrice Santorini and Mary Ann Marcinkiewicz. 1993. Building a Large Annotated Corpus of English: The Penn Treebank. *Computational Linguistics*, 19 (2): 313–330.

Marinov, Svetoslav. —. Parallel Dependency Treebank: A Case Study of 200 Swedish–Bulgarian Sentences. URL: `http://www.gslt.hum.gu.se/~svet/courses/mt2/parallelDT.pdf`

Matsumoto, Yuju, Hiroyuki Ishimoto and Takehito Utsuro. 1993. Structural Matching of Parallel Texts. In *31st Annual Meeting of the Association for Computational Linguistics (ACL '93)*, pp. 23–30. Columbus, OH.

Megyesi, Beáta, Bengt Dahlqvist, Eva Petersson and Joakim Nivre. 2008. Swedish–Turkish Parallel Treebank. In *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC '08)*. Marrakech, Morocco.

Menezes, Arul and Stephen D. Richardson. 2003. A Best-first Alignment Algorithm for Automatic Extraction of Transfer Mappings from Bilingual Corpora. In *Recent Advances in Example-Based Machine Translation* , eds. Michael Carl and Andy Way, chap. 15, pp. 421–442. Vol. 21 of *Text, Speech and Language Technology*. Dordrecht.

Menger, Karl. 1931. Bericht über ein Mathematisches Kolloquium. *Monatshefte für Mathematik und Physik* , 38: 17–38.

Meyers, Adam, Roman Yangarber and Ralph Grishman. 1996. Alignment of Shared Forests for Bilingual Corpora. In *Proceedings of the 16th International Conference on Computational Linguistics (CoLing '96)*. vol. 1, pp. 460–465. Copenhagen, Denmark.

Meyers, Adam, Roman Yangarber, Ralph Grishman, Catherine Macleod and Antonio Moreno-Sandoval. 1998. Deriving Transfer Rules from Dominance-Preserving Alignments. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and the 17th International Conference on Computational Linguistics (CoLing-ACL '98)*. vol. 2, pp. 843–847. Montreal, QC, Canada.

Müller, Christoph and Michael Strube. 2006. Multi-level annotation of linguistic data with MMAX2. In *Corpus Technology and Language Pedagogy. New Resources, New Tools, New Methods*, eds. Sabine Braun, Kurt Kohn and Joybrato Mukherjee, pp. 197–214. Vol. 3 of *English Corpus Linguistics*. Frankfurt, Germany: Peter Lang.

Nesson, Rebecca, Stuart M. Shieber and Alexander Rush. 2006. Induction of Probabilistic Synchronous Tree-Insertion Grammars for Machine Translation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas (AMTA '06)*, pp. 128–137. Cambridge, MA.

Nivre, Joakim, Johan Hall and Jens Nilsson. 2006. MaltParser: A Data-Driven Parser-Generator for Dependency Parsing. In *Proceedings of the Fifth International Conference on Language Resources and Evaluation (LREC '06)*, ed. Nicoletta Calzolari, pp. 2216–2219. Genoa, Italy.

Och, Franz Josef and Hermann Ney. 2003. A Systematic Comparison of Various Statistical Alignment Models. *Computational Linguistics*, 29 (1): 19–51.

Owczarzak, Karolina. 2008. A Novel Dependency-Based Evaluation Metric for Machine Translation. School of Computing, Dublin City University: PhD Thesis. Dublin, Ireland.

Papineni, Kishore, Salim Roukos, Todd Ward and Wei-Jing Zhu. 2002. BLEU: A Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association of Computational Linguistics (ACL '02)*, pp. 311–318. Philadelphia, PA.

Sadler, Victor and Ronald Vendelmans. 1990. Pilot Implementation of a Bilingual Knowledge Bank. In *Proceedings of the 13th International Conference on Computational Linguistics (CoLing '90)*, ed. Hans Karlgren. vol. 3, pp. 449–451. Helsinki, Finland.

Samuelsson, Yvonne and Martin Volk. 2006. Phrase Alignment in Parallel Treebanks. In *Proceedings of the 5th Workshop on Treebanks and Linguistic Theories (TLT '06)*, eds. Jan Hajič and Joakim Nivre, pp. 91–102. Prague, Czech Republic.

Samuelsson, Yvonne and Martin Volk. 2007. Alignment Tools for Parallel Treebanks. In *Data Structures for Linguistic Resources and Applications: Proceedings of the Biennial GLDV Conference 2007*, eds. Georg Rehm, Andreas Witt and Lothar Lemnitzer. Tübingen, Germany: Gunter Narr.

Schmid, Helmut. 2004. Efficient Parsing of Highly Ambiguous Context-Free Grammars with Bit Vectors. In *Proceedings of the 20th International Conference on Computational Linguistics (CoLing '04)*, pp. 162–168. Geneva, Switzerland.

Shieber, Stuart M. 1994. Restricting the Weak-Generative Capacity of Synchronous Tree-Adjoining Grammars. *Computational Intelligence* , 10 (4): 371–385.

Tinsley, John, Ventsislav Zhechev, Mary Hearne and Andy Way. 2007. Robust Language Pair-Independent Sub-Tree Alignment. In *Proceedings of the Machine Translation Summit XI*, ed. Bente Maegaard, pp. 467–474. Copenhagen, Denmark.

Tinsley, John, Mary Hearne and Andy Way. 2009. Parallel Treebanks in Phrase-Based Statistical Machine Translation. In *Computational Linguistics and Intellegent Text Processing: 10th International Conference (CICLing '09)*, ed. Alexander F. Gelbukh, pp. 318–331. Vol. 5449 of *Lecture Notes in Computer Science* . Mexico City, Mexico: Springer.

Uchimoto, Kiyotaka, Yujie Zhang, Kiyoshi Sudo, Masaki Murata, Satoshi Sekine and Hitoshi Isahara. 2004. Multilingual Aligned Parallel Treebank Corpus Reflecting Contextual Information and its Applications. In *Proceedings of the Workshop on Multilingual Linguistic Resources (MLR '04)*, eds. Gilles Sérasset, Susan Armstrong, Christian Boitet, Andrei Popescu-Belis and Dan Tufis, pp. 63–70. Geneva, Switzerland.

Uibo, Heli, Krista Liin and Martin Volk. 2005. Phrase alignment of Estonian–German parallel treebanks. Paper presented at *JRC EU-Enlargement Workshop 'Exploiting Parallel Corpora in up to 20 Languages'*, Arona, Italy.

Venugopal, Ashish and Andreas Zollmann. 2009. Grammar based statistical MT on Hadoop: An end-to-end toolkit for large scale PSCFG based MT. *The Prague Bulletin of Mathematical Linguistics*, 91: 67–78.

Viterbi, Andrew J. 1967. Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm. *IEEE Transactions on Information Theory*, 13 (2): 260–269.

Wellington, Benjamin, Sonjia Waxmonsky and I. Dan Melamed. 2006. Empirical Lower Bounds on the Complexity of Translational Equivalence. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (CoLing–ACL '06)*, eds. Nicoletta Calzolari, Claire Cardie and Pierre Isabelle, pp. 977–984. Sydney, Australia.

Whitney, Hassler. 1932. Congruent graphs and the connectivity of graphs. *American Journal of Mathematics*, 54: 150–168. [reprinted in: Hassler Whitney Collected Works, vol. 1 (eds. James Eells and Domingo Toledo), 1992, Boston, MA: Birkhäuser, pp. 61–79].

Wu, Dekai. 1995. An Algorithm for Simultaneously Bracketing Parallel Texts by Aligning Words. In *33rd Annual Meeting of the Association for Computational Linguistics (ACL '95): Proceedings of the Conference* , pp. 244–251. Cambridge, MA.

Wu, Dekai. 1997. Stochastic Inversion Transduction Grammars and Bilingual Parsing of Parallel Corpora. *Computational Linguistics*, 23 (3): 377–403.

Wu, Dekai. 2000. Bracketing and aligning words and constituents in parallel text using Stochastic Inversion Transduction Grammars. In *Parallel Text Processing: Alignment and Use of Translation Corpora*, ed. Jean Veronis, chap. 7, pp. 139–167. Dordrecht: Kluwer.

Younger, Daniel H. 1967. Recognition and Parsing of Context-Free Languages in Time n^3. *Information and Control*, 10 (2): 189–208.

Zhechev, Ventsislav and Andy Way. 2008. Automatic Generation of Parallel Treebanks. In *Proceedings of the 22nd International Conference on Computational Linguistics (CoLing '08)*, pp. 1105–1112. Manchester, UK.

Zhechev, Ventsislav. 2009. Unsupervised Generation of Parallel Treebanks through Sub-Tree Alignment. *The Prague Bulletin of Mathematical Linguistics*, 91: 89–98.

Appendix A Analysis Data Tables

In this appendix, you can find some tables containing raw data for the analysis of the properties of the various parallel treebanks discussed in this work.

System/ Configuration	Number of POS tags per data set			
	HomeCentre		EuroParl	
	English	French	English	German
CMU_TnT	91	94		
man_gold	91	94		
str2str_*	53	55	140	99
str2tree_*	53	94	140	150
tree2str_*	91	55	195	99
tree2tree_*	91	94	195	150

Table 1: Number of unique Part-of-Speech tags for the HomeCentre and EuroParl data sets

System/Configuration	Tag pairs	System/Configuration	Tag pairs
CMU_TnT	352	man_gold	241
str2str_11_gs	379	tree2str_11_gs	426
str2str_11r_gs	361	tree2str_11r_gs	364
str2str_111_gs	375	tree2str_111_gs	412
str2str_111r_gs	352	tree2str_111r_gs	331
str2str_12_gs	324	tree2str_12_gs	400
str2str_12r_gs	326	tree2str_12r_gs	384
str2str_121_gs	333	tree2str_121_gs	400
str2str_121r_gs	325	tree2str_121r_gs	361
str2str_21_gs	379	tree2str_21_gs	418
str2str_21r_gs	362	tree2str_21r_gs	360
str2str_211_gs	374	tree2str_211_gs	407
str2str_211r_gs	350	tree2str_211r_gs	329
str2str_22_gs	324	tree2str_22_gs	398
str2str_22r_gs	325	tree2str_22r_gs	382
str2str_221_gs	333	tree2str_221_gs	397
str2str_221r_gs	322	tree2str_221r_gs	359
str2tree_11_gs	430	tree2tree_11_gs	569
str2tree_11r_gs	374	tree2tree_11r_gs	539
str2tree_111_gs	421	tree2tree_111_gs	584
str2tree_111r_gs	352	tree2tree_111r_gs	563
str2tree_12_gs	403	tree2tree_12_gs	587
str2tree_12r_gs	365	tree2tree_12r_gs	555
str2tree_121_gs	411	tree2tree_121_gs	592
str2tree_121r_gs	356	tree2tree_121r_gs	562
str2tree_21_gs	421	tree2tree_21_gs	549
str2tree_21r_gs	368	tree2tree_21r_gs	529
str2tree_211_gs	414	tree2tree_211_gs	574
str2tree_211r_gs	343	tree2tree_211r_gs	560
str2tree_22_gs	398	tree2tree_22_gs	582
str2tree_22r_gs	361	tree2tree_22r_gs	556
str2tree_221_gs	408	tree2tree_221_gs	585
str2tree_221r_gs	354	tree2tree_221r_gs	560

Table 2: Number of unique source ↔ target pairs of linked Part-of-Speech tags in the HomeCentre data set

System/Configuration	tag pairs	System/Configuration	tag pairs
str2str_11_gs	2138	tree2str_11_gs	1719
str2str_111_gs	2123	tree2str_111_gs	1325
str2str_12_gs	1838	tree2str_12_gs	2121
str2str_121_gs	1937	tree2str_121_gs	2103
str2str_21_gs	2132	tree2str_21_gs	1700
str2str_211_gs	2122	tree2str_211_gs	1307
str2str_22_gs	1838	tree2str_22_gs	2087
str2str_221_gs	1939	tree2str_221_gs	2081
str2tree_11_gs	1975	tree2tree_11_gs	2717
str2tree_111_gs	1761	tree2tree_111_gs	2698
str2tree_12_gs	2127	tree2tree_12_gs	2711
str2tree_121_gs	2009	tree2tree_121_gs	2741
str2tree_21_gs	1955	tree2tree_21_gs	2673
str2tree_211_gs	1758	tree2tree_211_gs	2637
str2tree_22_gs	2076	tree2tree_22_gs	2663
str2tree_221_gs	1983	tree2tree_221_gs	2692

Table 3: Number of unique source ↔ target pairs of linked Part-of-Speech tags in the EuroParl data set

System/ Configuration	% linked source nodes	% linked target nodes	System/ Configuration	% linked source nodes	% linked target nodes
CMU_TnT	60.22%	53.96%	man_gold	62.14%	55.44%
str2str_11_gs	90.35%	81.51%	tree2str_11_gs	81.09%	79.57%
str2str_11r_gs	88.87%	80.12%	tree2str_11r_gs	71.80%	71.41%
str2str_111_gs	90.87%	82.37%	tree2str_111_gs	84.35%	80.22%
str2str_111r_gs	87.00%	78.44%	tree2str_111r_gs	73.25%	70.42%
str2str_12_gs	86.81%	78.05%	tree2str_12_gs	81.03%	76.29%
str2str_12r_gs	86.03%	77.10%	tree2str_12r_gs	72.87%	70.00%
str2str_121_gs	88.57%	79.66%	tree2str_121_gs	82.73%	77.40%
str2str_121r_gs	84.05%	75.18%	tree2str_121r_gs	74.87%	70.84%
str2str_21_gs	90.16%	81.38%	tree2str_21_gs	81.08%	79.20%
str2str_21r_gs	88.53%	79.72%	tree2str_21r_gs	71.66%	71.22%
str2str_211_gs	90.79%	82.28%	tree2str_211_gs	84.11%	79.95%
str2str_211r_gs	86.52%	77.99%	tree2str_211r_gs	73.19%	70.33%
str2str_22_gs	86.75%	77.96%	tree2str_22_gs	80.96%	76.17%
str2str_22r_gs	85.96%	77.05%	tree2str_22r_gs	72.81%	69.89%
str2str_221_gs	88.50%	79.57%	tree2str_221_gs	82.63%	77.27%
str2str_221r_gs	83.77%	74.94%	tree2str_221r_gs	74.73%	70.69%
str2tree_11_gs	89.26%	71.38%	tree2tree_11_gs	77.88%	68.23%
str2tree_11r_gs	80.05%	62.83%	tree2tree_11r_gs	75.75%	66.48%
str2tree_111_gs	89.22%	73.86%	tree2tree_111_gs	80.31%	70.26%
str2tree_111r_gs	78.80%	63.97%	tree2tree_111r_gs	77.81%	68.22%
str2tree_12_gs	85.11%	70.60%	tree2tree_12_gs	78.03%	68.27%
str2tree_12r_gs	77.69%	62.88%	tree2tree_12r_gs	75.68%	66.32%
str2tree_121_gs	86.65%	72.46%	tree2tree_121_gs	79.18%	69.31%
str2tree_121r_gs	78.91%	64.93%	tree2tree_121r_gs	76.97%	67.51%
str2tree_21_gs	88.89%	71.34%	tree2tree_21_gs	77.55%	67.89%
str2tree_21r_gs	79.90%	62.81%	tree2tree_21r_gs	75.52%	66.26%
str2tree_211_gs	89.01%	73.73%	tree2tree_211_gs	79.99%	69.96%
str2tree_211r_gs	78.56%	63.79%	tree2tree_211r_gs	77.66%	68.09%
str2tree_22_gs	84.88%	70.45%	tree2tree_22_gs	77.78%	68.04%
str2tree_22r_gs	77.45%	62.68%	tree2tree_22r_gs	75.51%	66.16%
str2tree_221_gs	86.49%	72.36%	tree2tree_221_gs	78.96%	69.11%
str2tree_221r_gs	78.72%	64.77%	tree2tree_221r_gs	76.78%	67.34%

Table 4: Average percentage of linked nodes in source and target trees in the HomeCentre data set

System/ Configuration	% linked source nodes	% linked target nodes	System/ Configuration	% linked source nodes	% linked target nodes
str2str_11_gs	83.66%	88.88%	tree2str_11_gs	29.01%	42.21%
str2str_111_gs	84.35%	88.77%	tree2str_111_gs	21.18%	30.73%
str2str_12_gs	77.24%	81.80%	tree2str_12_gs	55.17%	67.90%
str2str_121_gs	79.89%	84.39%	tree2str_121_gs	53.77%	64.00%
str2str_21_gs	83.74%	88.91%	tree2str_21_gs	28.77%	41.72%
str2str_211_gs	84.33%	88.74%	tree2str_211_gs	21.11%	30.62%
str2str_22_gs	77.25%	81.82%	tree2str_22_gs	54.78%	67.32%
str2str_221_gs	79.90%	84.40%	tree2str_221_gs	53.50%	63.67%
str2tree_11_gs	53.06%	48.54%	tree2tree_11_gs	59.98%	71.26%
str2tree_111_gs	48.91%	47.37%	tree2tree_111_gs	63.05%	74.99%
str2tree_12_gs	57.94%	55.58%	tree2tree_12_gs	58.51%	69.54%
str2tree_121_gs	54.74%	54.70%	tree2tree_121_gs	60.73%	72.23%
str2tree_21_gs	52.59%	48.24%	tree2tree_21_gs	58.41%	69.36%
str2tree_211_gs	48.74%	47.18%	tree2tree_211_gs	62.04%	73.76%
str2tree_22_gs	57.12%	54.85%	tree2tree_22_gs	57.47%	68.25%
str2tree_221_gs	54.28%	54.24%	tree2tree_221_gs	59.85%	71.14%

Table 5: Average percentage of linked nodes in source and target trees in the EuroParl data set

English side of the treebank		French side of the treebank	
System/Configuration	# of nodes	System/Configuration	# of nodes
CMU_TnT	15.33	CMU_TnT	17.52
man_gold	15.33	man_gold	17.52
str2str_11_gs	14.72	str2str_11_gs	16.25
str2str_11r_gs	14.54	str2str_11r_gs	16.09
str2str_111_gs	15.33	str2str_111_gs	16.82
str2str_111r_gs	15.12	str2str_111r_gs	16.70
str2str_12_gs	15.18	str2str_12_gs	16.83
str2str_12r_gs	15.07	str2str_12r_gs	16.82
str2str_121_gs	15.28	str2str_121_gs	16.93
str2str_121r_gs	15.12	str2str_121r_gs	16.85
str2str_21_gs	14.80	str2str_21_gs	16.34
str2str_21r_gs	14.52	str2str_21r_gs	16.08
str2str_211_gs	15.31	str2str_211_gs	16.81
str2str_211r_gs	15.10	str2str_211r_gs	16.68
str2str_22_gs	15.18	str2str_22_gs	16.84
str2str_22r_gs	15.08	str2str_22r_gs	16.82
str2str_221_gs	15.28	str2str_221_gs	16.93
str2str_221r_gs	15.13	str2str_221r_gs	16.86
str2tree_11_gs	12.77	str2tree_*_gs	17.52
str2tree_11r_gs	12.20	tree2str_11_gs	14.23
str2tree_111_gs	13.65	tree2str_11r_gs	13.73
str2tree_111r_gs	12.92	tree2str_111_gs	15.11
str2tree_12_gs	13.62	tree2str_111r_gs	14.51
str2tree_12r_gs	12.96	tree2str_12_gs	15.32
str2tree_121_gs	13.80	tree2str_12r_gs	14.69
str2tree_121r_gs	13.27	tree2str_121_gs	15.45
str2tree_21_gs	12.85	tree2str_121r_gs	14.99
str2tree_21r_gs	12.24	tree2str_21_gs	14.32
str2tree_211_gs	13.65	tree2str_21r_gs	13.75
str2tree_211r_gs	12.91	tree2str_211_gs	15.10
str2tree_22_gs	13.61	tree2str_211r_gs	14.51
str2tree_22r_gs	12.95	tree2str_22_gs	15.33
str2tree_221_gs	13.80	tree2str_22r_gs	14.69
str2tree_221r_gs	13.27	tree2str_221_gs	15.45
tree2str_*_gs	15.33	tree2str_221r_gs	14.98
tree2tree_*_gs	15.33	tree2tree_*_gs	17.52

Table 6: Average number of nodes in source and target trees in the HomeCentre data set

English side of the treebank		German side of the treebank	
System/Configuration	# of nodes	System/Configuration	# of nodes
str2str_11_gs	29.35	str2str_11_gs	27.60
str2str_111_gs	31.46	str2str_111_gs	29.91
str2str_12_gs	30.92	str2str_12_gs	29.23
str2str_121_gs	31.42	str2str_121_gs	29.78
str2str_21_gs	29.48	str2str_21_gs	27.74
str2str_211_gs	31.45	str2str_211_gs	29.90
str2str_22_gs	30.92	str2str_22_gs	29.23
str2str_221_gs	31.42	str2str_221_gs	29.77
str2tree_11_gs	22.13	str2tree_*_gs	24.86
str2tree_111_gs	23.24	tree2str_11_gs	19.24
str2tree_12_gs	23.65	tree2str_111_gs	19.26
str2tree_121_gs	24.41	tree2str_12_gs	24.00
str2tree_21_gs	22.17	tree2str_121_gs	24.69
str2tree_211_gs	23.22	tree2str_21_gs	19.28
str2tree_22_gs	23.65	tree2str_211_gs	19.25
str2tree_221_gs	24.39	tree2str_22_gs	24.01
tree2str_*_gs	30.03	tree2str_221_gs	24.68
tree2tree_*_gs	30.03	tree2tree_*_gs	24.86

Table 7: Average number of nodes in source and target trees in the EuroParl data set

System/Configuration	# of links	System/Configuration	# of links
CMU_TnT	7.83	man_gold	8.28
str2str_11_gs	12.99	tree2str_11_gs	10.87
str2str_11r_gs	12.59	tree2str_11r_gs	9.03
str2str_111_gs	13.51	tree2str_111_gs	11.69
str2str_111r_gs	12.60	tree2str_111r_gs	9.31
str2str_12_gs	12.67	tree2str_12_gs	11.11
str2str_12r_gs	12.45	tree2str_12r_gs	9.44
str2str_121_gs	12.94	tree2str_121_gs	11.37
str2str_121r_gs	11.93	tree2str_121r_gs	9.74
str2str_21_gs	13.06	tree2str_21_gs	10.85
str2str_21r_gs	12.53	tree2str_21r_gs	9.01
str2str_211_gs	13.49	tree2str_211_gs	11.63
str2str_211r_gs	12.53	tree2str_211r_gs	9.30
str2str_22_gs	12.65	tree2str_22_gs	11.10
str2str_22r_gs	12.44	tree2str_22r_gs	9.42
str2str_221_gs	12.93	tree2str_221_gs	11.35
str2str_221r_gs	11.88	tree2str_221r_gs	9.71
str2tree_11_gs	10.97	tree2tree_11_gs	10.30
str2tree_11r_gs	8.99	tree2tree_11r_gs	9.93
str2tree_111_gs	11.74	tree2tree_111_gs	10.93
str2tree_111r_gs	9.26	tree2tree_111r_gs	10.40
str2tree_12_gs	11.03	tree2tree_12_gs	10.50
str2tree_12r_gs	9.22	tree2tree_12r_gs	10.03
str2tree_121_gs	11.38	tree2tree_121_gs	10.67
str2tree_121r_gs	9.59	tree2tree_121r_gs	10.22
str2tree_21_gs	10.99	tree2tree_21_gs	10.20
str2tree_21r_gs	9.00	tree2tree_21r_gs	9.87
str2tree_211_gs	11.71	tree2tree_211_gs	10.84
str2tree_211r_gs	9.21	tree2tree_211r_gs	10.37
str2tree_22_gs	10.99	tree2tree_22_gs	10.45
str2tree_22r_gs	9.18	tree2tree_22r_gs	9.99
str2tree_221_gs	11.36	tree2tree_221_gs	10.62
str2tree_221r_gs	9.53	tree2tree_221r_gs	10.17

Table 8: Average number of sub-sentential links per sentence pair in the HomeCentre data set

System/Configuration	# of links	System/Configuration	# of links
str2str_11_gs	24.43	tree2str_11_gs	8.15
str2str_111_gs	26.41	tree2str_111_gs	5.69
str2str_12_gs	23.72	tree2str_12_gs	16.21
str2str_121_gs	24.87	tree2str_121_gs	16.37
str2str_21_gs	24.56	tree2str_21_gs	8.07
str2str_211_gs	26.40	tree2str_211_gs	5.66
str2str_22_gs	23.73	tree2str_22_gs	16.07
str2str_221_gs	24.87	tree2str_221_gs	16.27
str2tree_11_gs	11.25	tree2tree_11_gs	17.30
str2tree_111_gs	10.85	tree2tree_111_gs	18.28
str2tree_12_gs	13.20	tree2tree_12_gs	16.81
str2tree_121_gs	13.03	tree2tree_121_gs	17.48
str2tree_21_gs	11.16	tree2tree_21_gs	16.71
str2tree_211_gs	10.79	tree2tree_211_gs	17.91
str2tree_22_gs	12.98	tree2tree_22_gs	16.43
str2tree_221_gs	12.88	tree2tree_221_gs	17.15

Table 9: Average number of sub-sentential links per sentence pair in the EuroParl data set

System/ Configuration	% non-lexical links	% lexical links	System/ Configuration	% non-lexical links	% lexical links
CMU_TnT	40.13%	59.87%	man_gold	48.21%	51.79%
str2str_11_gs	38.33%	61.67%	tree2str_11_gs	31.11%	68.89%
str2str_11r_gs	38.19%	61.81%	tree2str_11r_gs	31.57%	68.43%
str2str_111_gs	40.36%	59.64%	tree2str_111_gs	34.87%	65.13%
str2str_111r_gs	42.26%	57.74%	tree2str_111r_gs	37.14%	62.86%
str2str_12_gs	40.33%	59.67%	tree2str_12_gs	34.25%	65.75%
str2str_12r_gs	40.43%	59.57%	tree2str_12r_gs	30.41%	69.59%
str2str_121_gs	40.66%	59.34%	tree2str_121_gs	35.34%	64.66%
str2str_121r_gs	44.06%	55.94%	tree2str_121r_gs	35.81%	64.19%
str2str_21_gs	38.69%	61.31%	tree2str_21_gs	31.71%	68.29%
str2str_21r_gs	38.31%	61.69%	tree2str_21r_gs	31.81%	68.19%
str2str_211_gs	40.33%	59.67%	tree2str_211_gs	34.99%	65.01%
str2str_211r_gs	42.54%	57.46%	tree2str_211r_gs	37.24%	62.76%
str2str_22_gs	40.40%	59.60%	tree2str_22_gs	34.36%	65.64%
str2str_22r_gs	40.47%	59.53%	tree2str_22r_gs	30.51%	69.49%
str2str_221_gs	40.69%	59.31%	tree2str_221_gs	35.42%	64.58%
str2str_221r_gs	44.33%	55.67%	tree2str_221r_gs	35.96%	64.04%
str2tree_11_gs	31.59%	68.41%	tree2tree_11_gs	29.06%	70.94%
str2tree_11r_gs	31.50%	68.50%	tree2tree_11r_gs	28.71%	71.29%
str2tree_111_gs	35.23%	64.77%	tree2tree_111_gs	32.21%	67.79%
str2tree_111r_gs	37.24%	62.76%	tree2tree_111r_gs	32.45%	67.55%
str2tree_12_gs	34.45%	65.55%	tree2tree_12_gs	31.54%	68.46%
str2tree_12r_gs	31.29%	68.71%	tree2tree_12r_gs	31.84%	68.16%
str2tree_121_gs	35.74%	64.26%	tree2tree_121_gs	32.68%	67.32%
str2tree_121r_gs	36.73%	63.27%	tree2tree_121r_gs	33.10%	66.90%
str2tree_21_gs	32.20%	67.80%	tree2tree_21_gs	29.66%	70.34%
str2tree_21r_gs	31.94%	68.06%	tree2tree_21r_gs	28.95%	71.05%
str2tree_211_gs	35.33%	64.67%	tree2tree_211_gs	32.34%	67.66%
str2tree_211r_gs	37.44%	62.56%	tree2tree_211r_gs	32.49%	67.51%
str2tree_22_gs	34.58%	65.42%	tree2tree_22_gs	31.70%	68.30%
str2tree_22r_gs	31.34%	68.66%	tree2tree_22r_gs	31.95%	68.05%
str2tree_221_gs	35.83%	64.17%	tree2tree_221_gs	32.80%	67.20%
str2tree_221r_gs	36.85%	63.15%	tree2tree_221r_gs	33.22%	66.78%

Table 10: Average distribution of the non-lexical and lexical links per sentence pair in the HomeCentre data set

System/ Configuration	% non- lexical links	% lexical links	System/ Configuration	% non- lexical links	% lexical links
str2str_11_gs	41.96%	58.04%	tree2str_11_gs	28.50%	71.50%
str2str_111_gs	47.92%	52.08%	tree2str_111_gs	37.34%	62.66%
str2str_12_gs	47.17%	52.83%	tree2str_12_gs	28.64%	71.36%
str2str_121_gs	48.14%	51.86%	tree2str_121_gs	36.93%	63.07%
str2str_21_gs	42.33%	57.67%	tree2str_21_gs	29.10%	70.90%
str2str_211_gs	47.93%	52.07%	tree2str_211_gs	37.63%	62.37%
str2str_22_gs	47.18%	52.82%	tree2str_22_gs	28.91%	71.09%
str2str_221_gs	48.15%	51.85%	tree2str_221_gs	37.13%	62.87%
str2tree_11_gs	30.54%	69.46%	tree2tree_11_gs	14.41%	85.59%
str2tree_111_gs	42.95%	57.05%	tree2tree_111_gs	24.16%	75.84%
str2tree_12_gs	23.64%	76.36%	tree2tree_12_gs	19.58%	80.42%
str2tree_121_gs	37.99%	62.01%	tree2tree_121_gs	25.03%	74.97%
str2tree_21_gs	31.28%	68.72%	tree2tree_21_gs	15.22%	84.78%
str2tree_211_gs	43.17%	56.83%	tree2tree_211_gs	24.50%	75.50%
str2tree_22_gs	24.02%	75.98%	tree2tree_22_gs	19.99%	80.01%
str2tree_221_gs	38.30%	61.70%	tree2tree_221_gs	25.39%	74.61%

Table 11: Average distribution of the non-lexical and lexical links per sentence pair in the EuroParl data set

System/ Configuration	# of links	# of sentences	System/ Configuration	# of links	# of sentences
str2str_11_gs	4.79	80	tree2str_11_gs	5.88	84
str2str_11r_gs	5.51	65	tree2str_11r_gs	5.64	56
str2str_111_gs	4.23	77	tree2str_111_gs	5.18	55
str2str_111r_gs	10.99	70	tree2str_111r_gs	4.09	35
str2str_12_gs	4.52	117	tree2str_12_gs	5.28	69
str2str_12r_gs	4.44	124	tree2str_12r_gs	6.74	61
str2str_121_gs	4.24	121	tree2str_121_gs	4.46	67
str2str_121r_gs	9.59	133	tree2str_121r_gs	4.25	48
str2str_21_gs	6.52	89	tree2str_21_gs	8.72	119
str2str_21r_gs	7.46	83	tree2str_21r_gs	5.45	85
str2str_211_gs	5.80	91	tree2str_211_gs	10.85	79
str2str_211r_gs	13.12	83	tree2str_211r_gs	11.90	49
str2str_22_gs	4.70	123	tree2str_22_gs	6.17	78
str2str_22r_gs	5.17	131	tree2str_22r_gs	6.57	74
str2str_221_gs	4.39	125	tree2str_221_gs	5.51	77
str2str_221r_gs	10.38	139	tree2str_221r_gs	4.62	61
str2tree_11_gs	5.53	64	tree2tree_11_gs	6.62	94
str2tree_11r_gs	4.98	54	tree2tree_11r_gs	6.63	88
str2tree_111_gs	4.31	45	tree2tree_111_gs	5.37	79
str2tree_111r_gs	4.26	35	tree2tree_111r_gs	5.74	68
str2tree_12_gs	4.39	61	tree2tree_12_gs	6.93	98
str2tree_12r_gs	6.70	56	tree2tree_12r_gs	7.60	101
str2tree_121_gs	5.04	55	tree2tree_121_gs	6.77	98
str2tree_121r_gs	4.81	48	tree2tree_121r_gs	6.91	93
str2tree_21_gs	6.14	106	tree2tree_21_gs	8.14	155
str2tree_21r_gs	5.66	79	tree2tree_21r_gs	7.34	123
str2tree_211_gs	5.24	66	tree2tree_211_gs	6.19	113
str2tree_211r_gs	5.66	53	tree2tree_211r_gs	5.60	90
str2tree_22_gs	5.73	78	tree2tree_22_gs	7.21	124
str2tree_22r_gs	7.26	73	tree2tree_22r_gs	7.58	120
str2tree_221_gs	5.43	70	tree2tree_221_gs	7.07	121
str2tree_221r_gs	8.09	56	tree2tree_221r_gs	7.48	111

Table 12: Average number of undecided links per sentence pair and total number of sentences with undecided links per configuration for the HomeCentre data set

System/ Configuration	# of links	# of sentences	System/ Configuration	# of links	# of sentences
str2str_11_gs	4.49	379	tree2str_11_gs	5.84	256
str2str_111_gs	4.50	454	tree2str_111_gs	3.57	49
str2str_12_gs	5.30	1220	tree2str_12_gs	6.31	810
str2str_121_gs	5.07	1275	tree2str_121_gs	5.17	606
str2str_21_gs	5.79	514	tree2str_21_gs	11.54	1015
str2str_211_gs	6.20	530	tree2str_211_gs	16.72	290
str2str_22_gs	6.84	1207	tree2str_22_gs	8.53	1782
str2str_221_gs	6.81	1247	tree2str_221_gs	7.39	1261
str2tree_11_gs	6.48	227	tree2tree_11_gs	5.68	1166
str2tree_111_gs	4.90	62	tree2tree_111_gs	5.33	907
str2tree_12_gs	8.38	958	tree2tree_12_gs	10.04	2882
str2tree_121_gs	7.95	489	tree2tree_121_gs	8.96	2411
str2tree_21_gs	13.95	1223	tree2tree_21_gs	10.69	3900
str2tree_211_gs	16.07	522	tree2tree_211_gs	8.45	2758
str2tree_22_gs	10.97	2323	tree2tree_22_gs	13.09	4242
str2tree_221_gs	10.72	1453	tree2tree_221_gs	11.34	3723

Table 13: Average number of undecided links per sentence pair and total number of sentences with undecided links per configuration for the EuroParl data set

System/Configuration	# of reductions	System/Configuration	# of reductions
str2str_11_gs	13.17	tree2str_11_gs	11.31
str2str_11r_gs	14.08	tree2str_11r_gs	10.68
str2str_111_gs	14.72	tree2str_111_gs	13.06
str2str_111r_gs	14.21	tree2str_111r_gs	11.13
str2str_12_gs	13.62	tree2str_12_gs	12.12
str2str_12r_gs	13.68	tree2str_12r_gs	11.03
str2str_121_gs	14.11	tree2str_121_gs	12.77
str2str_121r_gs	13.18	tree2str_121r_gs	11.27
str2str_21_gs	13.23	tree2str_21_gs	11.29
str2str_21r_gs	12.27	tree2str_21r_gs	9.52
str2str_211_gs	14.69	tree2str_211_gs	12.99
str2str_211r_gs	12.76	tree2str_211r_gs	10.38
str2str_22_gs	13.60	tree2str_22_gs	12.10
str2str_22r_gs	12.32	tree2str_22r_gs	10.16
str2str_221_gs	14.09	tree2str_221_gs	12.74
str2str_221r_gs	11.99	tree2str_221r_gs	10.48
str2tree_11_gs	11.44	tree2tree_11_gs	10.87
str2tree_11r_gs	10.72	tree2tree_11r_gs	11.57
str2tree_111_gs	13.15	tree2tree_111_gs	12.39
str2tree_111r_gs	11.11	tree2tree_111r_gs	12.20
str2tree_12_gs	12.09	tree2tree_12_gs	11.57
str2tree_12r_gs	10.87	tree2tree_12r_gs	11.62
str2tree_121_gs	12.82	tree2tree_121_gs	12.11
str2tree_121r_gs	11.09	tree2tree_121r_gs	11.68
str2tree_21_gs	11.46	tree2tree_21_gs	10.78
str2tree_21r_gs	9.59	tree2tree_21r_gs	10.30
str2tree_211_gs	13.11	tree2tree_211_gs	12.30
str2tree_211r_gs	10.33	tree2tree_211r_gs	11.36
str2tree_22_gs	12.05	tree2tree_22_gs	11.51
str2tree_22r_gs	9.96	tree2tree_22r_gs	10.55
str2tree_221_gs	12.80	tree2tree_221_gs	12.06
str2tree_221r_gs	10.33	tree2tree_221r_gs	10.85

Table 14: Average number of search space reduction stages per pair for the HomeCentre data set

System/Configuration	# of reductions	System/Configuration	# of reductions
str2str_11_gs	25.47	tree2str_11_gs	10.05
str2str_111_gs	28.12	tree2str_111_gs	7.68
str2str_12_gs	24.88	tree2str_12_gs	18.07
str2str_121_gs	26.20	tree2str_121_gs	18.25
str2str_21_gs	25.60	tree2str_21_gs	9.97
str2str_211_gs	28.11	tree2str_211_gs	7.65
str2str_22_gs	24.89	tree2str_22_gs	17.93
str2str_221_gs	26.20	tree2str_221_gs	18.15
str2tree_11_gs	12.67	tree2tree_11_gs	19.08
str2tree_111_gs	12.83	tree2tree_111_gs	20.20
str2tree_12_gs	14.95	tree2tree_12_gs	18.59
str2tree_121_gs	14.95	tree2tree_121_gs	19.26
str2tree_21_gs	12.58	tree2tree_21_gs	18.49
str2tree_211_gs	12.77	tree2tree_211_gs	19.83
str2tree_22_gs	14.73	tree2tree_22_gs	18.21
str2tree_221_gs	14.81	tree2tree_221_gs	18.94

Table 15: Average number of search space reduction stages per pair for the EuroParl data set

Appendix B Evaluation Data Tables

In this appendix, you can find some tables containing raw data from the extrinsic evaluation of the various parallel treebanks discussed in this work.

System/ Configuration	BLEU	NIST	System/ Configuration	BLEU	NIST
CMU_TnT	0.5078	6.7739	man_gold	0.5285	7.0632
str2str_11_gs	0.5076	6.8919	tree2str_11_gs	0.5038	6.9178
str2str_11r_gs	0.4926	6.8582	tree2str_11r_gs	0.4960	6.7801
str2str_111_gs	0.5055	6.8226	tree2str_111_gs	0.5268	6.9533
str2str_111r_gs	0.5152	7.0346	tree2str_111r_gs	0.5023	6.8563
str2str_12_gs	0.5123	6.8741	tree2str_12_gs	0.5240	7.0161
str2str_12r_gs	0.5200	6.9120	tree2str_12r_gs	0.5155	6.9329
str2str_121_gs	0.5149	6.8887	tree2str_121_gs	0.5251	6.9665
str2str_121r_gs	0.5144	6.8612	tree2str_121r_gs	0.5200	6.9201
str2str_21_gs	0.5122	6.9484	tree2str_21_gs	0.5211	7.0244
str2str_21r_gs	0.4942	6.8522	tree2str_21r_gs	0.4896	6.7306
str2str_211_gs	0.5012	6.8185	tree2str_211_gs	0.5304	6.9737
str2str_211r_gs	0.5152	7.0359	tree2str_211r_gs	0.5028	6.8583
str2str_22_gs	0.5092	6.8398	tree2str_22_gs	0.5261	7.0278
str2str_22r_gs	0.5192	6.8971	tree2str_22r_gs	0.5125	6.9095
str2str_221_gs	0.5113	6.8595	tree2str_221_gs	0.5263	6.9747
str2str_221r_gs	0.5130	6.8177	tree2str_221r_gs	0.5224	6.9502
str2tree_11_gs	0.5154	6.9888	tree2tree_11_gs	0.5075	6.9228
str2tree_11r_gs	0.4771	6.6619	tree2tree_11r_gs	0.5052	6.8416
str2tree_111_gs	0.5278	6.9721	tree2tree_111_gs	0.5267	7.0041
str2tree_111r_gs	0.4948	6.7657	tree2tree_111r_gs	0.5232	6.9080
str2tree_12_gs	0.5252	7.0098	tree2tree_12_gs	0.5257	6.9704
str2tree_12r_gs	0.5044	6.9029	tree2tree_12r_gs	0.5186	6.9844
str2tree_121_gs	0.5326	7.0641	tree2tree_121_gs	0.5340	6.9675
str2tree_121r_gs	0.5201	6.9560	tree2tree_121r_gs	0.5271	7.0476
str2tree_21_gs	0.5137	6.9592	tree2tree_21_gs	0.5058	6.9624
str2tree_21r_gs	0.4834	6.7537	tree2tree_21r_gs	0.5087	6.8801
str2tree_211_gs	0.5248	6.9332	tree2tree_211_gs	0.5277	6.9996
str2tree_211r_gs	0.4951	6.7742	tree2tree_211r_gs	0.5242	6.9463
str2tree_22_gs	0.5226	6.9805	tree2tree_22_gs	0.5261	6.9791
str2tree_22r_gs	0.5044	6.9144	tree2tree_22r_gs	0.5158	6.9560
str2tree_221_gs	0.5308	7.0446	tree2tree_221_gs	0.5339	7.0044
str2tree_221r_gs	0.5174	6.9520	tree2tree_221r_gs	0.5233	7.0217

Table 1: BLEU and NIST scores from the evaluation of English–to–French translation using various versions of the HomeCentre parallel treebank

System/ Configuration	METEOR	Coverage	System/ Configuration	METEOR	Coverage
CMU_TnT	68.52%	62.92%	man_gold	72.67%	68.54%
str2str_11_gs	72.27%	92.29%	tree2str_11_gs	71.98%	75.63%
str2str_11r_gs	71.01%	92.29%	tree2str_11r_gs	70.36%	71.67%
str2str_111_gs	72.22%	91.25%	tree2str_111_gs	73.21%	78.13%
str2str_111r_gs	72.62%	89.58%	tree2str_111r_gs	71.52%	71.88%
str2str_12_gs	72.43%	92.50%	tree2str_12_gs	72.86%	77.29%
str2str_12r_gs	72.97%	90.83%	tree2str_12r_gs	71.99%	72.50%
str2str_121_gs	73.01%	92.29%	tree2str_121_gs	72.85%	78.33%
str2str_121r_gs	72.53%	88.13%	tree2str_121r_gs	71.59%	74.17%
str2str_21_gs	72.50%	92.92%	tree2str_21_gs	72.92%	75.63%
str2str_21r_gs	71.30%	91.88%	tree2str_21r_gs	70.04%	71.46%
str2str_211_gs	71.96%	91.25%	tree2str_211_gs	73.43%	78.13%
str2str_211r_gs	72.54%	89.38%	tree2str_211r_gs	71.58%	71.67%
str2str_22_gs	72.35%	92.29%	tree2str_22_gs	73.12%	77.29%
str2str_22r_gs	73.08%	90.83%	tree2str_22r_gs	71.72%	72.50%
str2str_221_gs	73.27%	92.29%	tree2str_221_gs	72.96%	78.33%
str2str_221r_gs	72.25%	87.71%	tree2str_221r_gs	71.72%	73.75%
str2tree_11_gs	71.65%	80.00%	tree2tree_11_gs	71.29%	71.67%
str2tree_11r_gs	69.89%	74.38%	tree2tree_11r_gs	71.18%	69.38%
str2tree_111_gs	72.88%	83.96%	tree2tree_111_gs	72.77%	72.50%
str2tree_111r_gs	70.52%	72.08%	tree2tree_111r_gs	72.64%	70.21%
str2tree_12_gs	72.53%	78.96%	tree2tree_12_gs	72.99%	72.08%
str2tree_12r_gs	70.80%	72.08%	tree2tree_12r_gs	71.82%	71.04%
str2tree_121_gs	73.28%	82.50%	tree2tree_121_gs	73.76%	72.08%
str2tree_121r_gs	71.74%	75.00%	tree2tree_121r_gs	72.74%	71.67%
str2tree_21_gs	72.34%	79.38%	tree2tree_21_gs	71.36%	71.25%
str2tree_21r_gs	70.40%	73.96%	tree2tree_21r_gs	71.51%	69.58%
str2tree_211_gs	72.73%	83.96%	tree2tree_211_gs	72.70%	72.50%
str2tree_211r_gs	70.58%	72.08%	tree2tree_211r_gs	72.79%	70.21%
str2tree_22_gs	72.16%	78.96%	tree2tree_22_gs	72.89%	71.88%
str2tree_22r_gs	70.85%	72.29%	tree2tree_22r_gs	71.45%	71.04%
str2tree_221_gs	73.11%	82.08%	tree2tree_221_gs	74.00%	71.88%
str2tree_221r_gs	71.44%	75.00%	tree2tree_221r_gs	72.44%	71.25%

Table 2: METEOR and coverage scores from the evaluation of English–to–French translation using various versions of the HomeCentre parallel treebank

System/ Configuration	BLEU	NIST	System/ Configuration	BLEU	NIST
str2tree_11_gs	0.0318	1.1280	tree2str_21_gs	0.0312	1.0732
str2tree_111_gs	0.0117	0.7437	tree2str_211_gs	0.0132	0.7424
str2tree_12_gs	0.1211	2.9097	tree2str_22_gs	0.1203	3.1886
str2tree_121_gs	0.0627	2.0401	tree2str_221_gs	0.1289	3.0685
str2tree_21_gs	0.0317	1.1357	tree2tree_11_gs	0.1215	3.0921
str2tree_211_gs	0.0122	0.7394	tree2tree_111_gs	0.1031	2.6339
str2tree_22_gs	0.1053	2.7542	tree2tree_12_gs	0.1318	3.0599
str2tree_221_gs	0.0249	1.0827	tree2tree_121_gs	0.1209	2.7943
tree2str_11_gs	0.0372	1.2731	tree2tree_21_gs	0.1092	2.8608
tree2str_111_gs	0.0129	0.7342	tree2tree_211_gs	0.1013	2.5739
tree2str_12_gs	0.1226	3.2462	tree2tree_22_gs	0.1249	3.0720
tree2str_121_gs	0.1237	3.0898	tree2tree_221_gs	0.1213	2.7998

Table 3: BLEU and NIST scores from the evaluation of English–to–German translation using various versions of the EuroParl parallel treebank

System/ Configuration	METEOR	Coverage	System/ Configuration	METEOR	Coverage
str2tree_11_gs	9.90%	80.32%	tree2str_21_gs	10.14%	50.31%
str2tree_111_gs	5.88%	75.62%	tree2str_211_gs	5.11%	28.67%
str2tree_12_gs	38.36%	83.52%	tree2str_22_gs	36.31%	84.06%
str2tree_121_gs	21.07%	79.33%	tree2str_221_gs	33.75%	76.51%
str2tree_21_gs	10.16%	79.98%	tree2tree_11_gs	32.44%	37.93%
str2tree_211_gs	5.94%	75.62%	tree2tree_111_gs	32.71%	65.74%
str2tree_22_gs	35.08%	84.30%	tree2tree_12_gs	34.88%	50.23%
str2tree_221_gs	9.93%	84.48%	tree2tree_121_gs	35.46%	67.98%
tree2str_11_gs	11.77%	55.85%	tree2tree_21_gs	31.80%	39.59%
tree2str_111_gs	5.19%	29.00%	tree2tree_211_gs	32.40%	65.74%
tree2str_12_gs	36.88%	84.06%	tree2tree_22_gs	35.09%	49.39%
tree2str_121_gs	33.53%	75.28%	tree2tree_221_gs	35.70%	67.29%

Table 4: METEOR and coverage scores from the evaluation of English–to–German translation using various versions of the EuroParl parallel treebank