1 Introduction

In this paper, we describe the methods and resources used to build the *FinnTreeBank-3* (FTB-3) parsebank, a 76.4 million token corpus of Finnish with automatically produced morphological and dependency syntax analyses. The corpus is a resource developed within the FIN-CLARIN consortium, the Finnish member of the CLARIN infrastructure project[1] and aims at supporting research and language technology development requiring large-scale parsed corpora. Further, as the underlying texts consist of the multilingual parallel corpora EuroParl (Koehn, 2005) and JRC-Acquis (Steinberger et al., 2006), corresponding parsebanks can be constructed for a number of other languages into which these two corpora have been translated as well. The larger context of the FTB-3 parsebank is described by Voutilainen et al. (2012b); our involvement in its development was through a public request for quotation issued by FIN-CLARIN, seeking the development of a sufficiently accurate Finnish syntactic parser and its application to the EuroParl and JRC-Acquis corpora. Our starting point for the development was thus untypical as the corpus text, morphological tagset, as well as the dependency scheme were all defined by FIN-CLARIN and not negotiable. Our sole task was to develop a parsing pipeline with sufficient accuracy and produce the actual parsebank data, in a contract research setting. Therefore, rather than being used as-is, all tools and resources at our disposal had to be adjusted so as to conform to the specifications of the project.

In the following sections, we will summarize the tools and resources used when developing FTB-3 as well as their adaptation to the target scheme and text corpus. Then, we will present the parsing pipeline and the evaluation of the resulting parsebank.

2 Available tools and resources

The parsing process consists of two major steps: morphological tagging and dependency parsing. Morphological tagging is carried out using an adapted version of *FinCG*, a commercial morphological tagger by Lingsoft, Inc.[2] The adaptations of FinCG specific to FTB-3 development are described in Section 4.1. Unlike for tagging, no pre-existing tools were at our disposal for dependency parsing. It was thus necessary to train a statistical dependency parser which, in turn, requires a suitable treebank that is annotated in the target dependency scheme.

For Finnish, there are two manually annotated treebanks available: the *Turku Dependency Treebank (TDT)* (Haverinen et al., 2010, 2011) and *FinnTreeBank (FTB)*, in its first version *FTB-1* (Voutilainen et al., 2011) when this work was carried out. The treebanks are developed for different purposes and are in many respects complementary. TDT has been specifically developed to support statistical parser training and consists of texts from various genres and text sources, aiming to serve as a representative selection of general Finnish. FTB-1, on the other hand, was developed within FIN-CLARIN as a grammar definition treebank. Its underlying corpus comprises all grammar examples from the Finnish grammar reference book of Hakulinen et al. (2004), in total 162,312 tokens in 19,140 examples. Together with its accompanying annotation manual (Voutilainen et al., 2012a), FTB-1 serves as the definition of the target dependency scheme for FTB-3. While, by its nature, it exhibits a wide variety of grammatical phenomena, this corpus of carefully selected grammar examples is not intended for statistical parser training as it does not have the same distributional properties as general Finnish text.

We based the statistical parser used in this work on TDT, as the treebank is more suitable

[1] `http://www.clarin.eu`
[2] `http://www.lingsoft.fi/`

Dependency type	Description
main	main predicate of the sentence
aux	auxiliary
subj	subject
obj	object
scomp	predicative
advl	adverbial
attr	attribute
phrm	phrase marker (conjunctions, adpositions etc.)
modal	the nominal part of a verb chain
phrv	phrasal verb
comp	comparison structure
idiom	idiom
conjunct	conjunct, coordination
voc	vocative
mod	post-modifier

Table 1: Dependency types of the FTB scheme.

for statistical parser training. To further improve the applicability of the treebank to the development of FTB-3, we manually annotated additional data from the EuroParl (19,964 tokens in 1,082 sentences) and JRC Acquis (24,909 tokens in 1,141 sentences) corpora, resulting in the final training data size of 190,271 tokens in 13,997 sentences.

3 Dependency scheme transformation

TDT is annotated in a slightly modified version of the widely used *Stanford Dependencies (SD)* scheme (de Marneffe and Manning, 2008b,a; Haverinen, 2012) which differs notably from the target scheme of FTB-3. The annotation of the treebank thus needs to be transformed to conform to the target scheme.

The two schemes differ both in the dependency types they define, as well as in the structure of the dependency trees for a number of important phenomena. While the SD scheme, as used in TDT, defines a total of 46 dependency types, the FTB scheme defines a considerably smaller set of 15 types, listed in Table 1. The tree structures notably differ as well, with 19.8% of the tokens in the target (FTB) trees being governed by a different token than in the source (SD) trees. The transformation therefore involves both dependency type mapping and modification of the governor–dependent relation. The transformation is carried out using a hybrid system consisting of hand-written rules, followed by a machine learning component that finalizes the trees by connecting islands resulting from incomplete rule coverage.

3.1 Transformation rules

Each transformation rule matches an arbitrarily complex pattern in the source tree and produces a single dependency in the target tree. The rule definition syntax allows restrictions on token text, lemmas and morphological tags, as well as dependency types and directions. Further, any restriction can also be negated, requiring that it must not be met for the pattern to match.

A typical rule transforms one SD dependency into the FTB scheme. Multiple additional constraints limiting the rule application to the correct context are typical as the two schemes treat differently several important structures. For instance, while in the SD scheme the subject in a

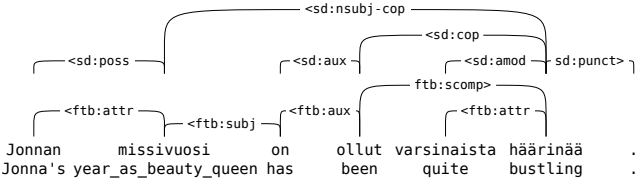

Figure 1: Example trees in the SD and FTB scheme for copula constructs.

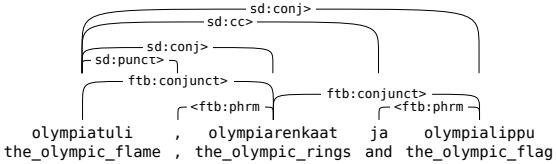

Figure 2: Example trees in the SD and FTB scheme for coordination.

copula construct is governed by the predicative, in the FTB scheme the subject is governed by the copular verb or, if present, the auxiliary (see Figure 1). Similarly, if a noun is preceded by several modifiers, the SD scheme attaches them all to the noun whereas the FTB chains them and only attaches the closest modifier to the noun. In addition, as the schemes often differ in parts of speech assignment in borderline cases, every rule possibly concerning for instance auxiliaries or adpositions must take into account the differences between the definitions of these groups of words.

As an example, the rule

$$\text{dep}(\text{amod}, T_1, T_2) \textit{ and not } \text{dep}(\text{amod}|\text{poss}|\text{num}, T_1, T_x) \rightarrow \text{dep}(\text{attr}, T_1, T_2)$$

describes the transformation of an adjectival modifier dependency (*amod*) to an attribute dependency (*attr*), with the token T_1 governing the token T_2. The negated restriction indicates that the rule should not be applied if T_1 also has another dependent with the dependency type *amod*, *poss* or *num*. This is because in such a case, the FTB scheme chains the modifiers and only attaches the closest one to the noun.

As a second example, the rule

$$\text{dep}(\text{xcomp}, T_1, T_2) \textit{ and } \text{is-ftb-aux}(T_2) \textit{ and not } \text{dep}(\text{cop}, T_1, T_x) \rightarrow \text{dep}(\text{aux}, T_2, T_1)$$

describes the transformation of a clausal complement dependency *xcomp* between T_1 and T_2 to an *aux* dependency between T_2 and T_1, for a verb that is defined as an auxiliary in the FTB scheme. For this, the rule is delimited to apply only when the T_2 token belongs to a group of lemmas including all the auxiliaries in the FTB scheme. The negation declares that the rule should not be applied if T_1 governs a *cop* dependency, meaning that it actually is a predicative. In this case the FTB scheme assigns the copular verb as the head (see Figure 1).

A specific challenge is posed by the transformation of coordination structures, shown in Figure 2, which can consist of arbitrarily many coordinated elements and are thus not easily addressed by the rules. Coordination structures are therefore transformed separately, by a dedicated program. A second case transformed by a dedicated program rather than by the rules are the

null tokens which in TDT represent the missing head token in gapping structures and fragments. The target FTB scheme, on the other hand, does not allow null tokens. As the last step of the transformation, we thus remove these tokens from TDT, selecting one of their dependents to act as the new governor using a priority list of dependency types and re-attaching all other dependents to the new governor.

3.2 Development of the transformation rules

To facilitate the development of the transformation rules, we have manually annotated the SD scheme trees for 17,061 tokens (1,992 sentences) from FinnTreeBank-1. These nearly 2,000 sentences thus have their syntax available in both schemes and can serve as test data in rule development.

The rules were developed iteratively in a GUI application designed for the purpose. The application presents the source and target trees, using color-coding to distinguish correct, missing, and extraneous dependencies in the transformation output. Further, the application allows to search among all sentences for arbitrary patterns, with the same expressive power as the rule pattern matching, both on the source and target side.

In total, the transformation ruleset consists of 305 rules. The rules are applied independently of each other, that is, all rules are tested and applied to all matching positions in the tree. Out of all trees, 83.3% are transformed in their entirety, resulting in a tree in the target scheme. Further 10.0% are transformed partially, resulting in several disconnected islands which are themselves trees. Such partial transformation results from the inevitable incomplete coverage of the transformation rules. Finally, the transformation output for the remaining 6.8% of trees contains a structure violating treeness: either a token with two governors or a cycle. These erroneous structures can be attributed to rule conflicts. We therefore post-process the transformation output using a machine learning component which connects the islands into complete trees as well as corrects erroneous structures by removing extra dependencies. This component is described in the following section.

3.3 Transformation post-processing using machine learning

The post-processing involves two distinct operations: removal of extraneous dependencies and insertion of new dependencies so as to connect islands in the analysis. We approach both of these subtasks using a classifier which, given any two tokens T_1 and T_2, returns the score of the most likely dependency type for the hypothetical $T_1 \rightarrow T_2$ dependency, and the score of there being no dependency with T_1 governing T_2. As the underlying machine learning algorithm for the classifier, we apply the regularized least squares ranker implemented in the RLScore package of Pahikkala et al. (2007).

We use several distinct types of features in the classification. *Token features* are generated separately for T_1 and T_2 and include the token itself, its lemma, and a binary feature for each of its morphological tags. *Target tree features* are extracted separately for T_1 and T_2 from the target, i.e. transformed, tree and consist of the governor type and all dependent types of the token in question. If there is no governor, or there are no dependents, a binary feature encoding this information is issued instead. Further, a binary feature encodes whether the token in question is a left or right dependent in the linear order of the sentence. A final group of features are the *source tree features* which capture the syntactic relationship between the two tokens in the source (SD) tree. If there is a source-tree dependency between T_1 and T_2, concatenation of its

type and direction (i.e. whether the dependency is $T_1 \rightarrow T_2$ or $T_2 \rightarrow T_1$ regardless of the linear order of the tokens in the sentence) are given as a feature. If there is no such direct dependency, a "middle man" token T_x is searched such that it links T_1 and T_2, regardless of dependency directions. If found, features encoding the type and direction of the connection between T_x and T_1 and T_x and T_2 are generated, as well as a feature encoding the entire path from T_1 to T_2 via T_x. If no such interconnecting token is found, a feature is generated encoding this fact.

The classifier is trained on the treebank transformed using the 305 rules described previously. A positive example is generated from every dependency in the target tree. We cannot, however, assume that any dependency not present in the target tree constitutes a valid negative example. First, these include dependencies that should have been generated by the rules and the ranker thus should specifically not be given these cases among the negative examples. Second, producing a negative example from any pair of tokens unconnected in the target tree would result in a large number of irrelevant negative examples of tokens that are in no way related to each other. Therefore, for every dependency $T_1 \rightarrow T_2$ in the tree, we produce a negative example from $g(T_1) \rightarrow T_2$ and every $d(T_1) \rightarrow T_2$ where $g(T_1)$ and $d(T_1)$ refer to the governor and dependents of T_1. In this way, more plausible negative examples are generated, between tokens that are more closely, even though not directly, related.

The first step in the post-processing is the removal of extraneous dependencies. Whenever a token has several governors, only the dependency with the highest score as given by the classifier is preserved, all others are removed. Directed cycles are broken by removing the dependency with the lowest score in the cycle.

In the second step, we connect islands in the analysis using a simple greedy algorithm. Note that the first post-processing step guarantees that each of the islands is itself a tree. First, we generate the set of all token pairs (T_1, T_2) such that T_1 and T_2 belong to different islands and T_2 is the root of the island to which it belongs. These pairs represent all hypothetical dependencies $T_1 \rightarrow T_2$ that can be, individually, inserted into the analysis without violating its treeness. Then we use the classifier to obtain the most likely dependency type and its score for each of these token pairs, even in cases where the prediction of there being no dependency had a higher score. Finally, progressing through the list of candidate dependencies ranked by their score, we insert each dependency if and only if it would not violate the treeness constraints, taking into account also the dependencies inserted so far.

This completes the description of the transformation of TDT into the target FTB dependency scheme. We now turn to describe the dependency parsing pipeline, trained on the transformed treebank.

4 Parsing pipeline

The dependency parsing pipeline comprises of a sentence splitter, tokenizer, morphological tagger, and statistical dependency parser.

4.1 Morphological tagging

Tokenization, sentence splitting, and morphological tagging were carried out using the commercial *FinCG* tagger and associated tools developed by Lingsoft, Inc.[3] The target morphological tagset was given as part of the FTB-3 specification, however, unlike for dependency syntax

[3] `http://www.lingsoft.fi`

no specification was given regarding the preferred analysis of borderline wordforms which can be analyzed in several equally plausible ways (Voutilainen et al., 2012b). Therefore, adapting FinCG to the target scheme could be implemented via a mapping table from FinCG's morphological tagset to the target tagset and did not require adjustments to the lexicon.

The underlying text of the FTB-3 corpus is comprised of European Union legal and parliamentary texts, and contains a number of domain-specific words not included in the general-purpose FinCG lexicon. Such unrecognized words are not given any lexical analysis by FinCG and their proportion in text must be kept to a minimum. The general-purpose FinCG lexicon was therefore augmented with Lingsoft's proprietary EU style checker lexicon and with a special domain lexicon for commercial vocabulary. The lexicon was thereafter further extended with a number of additional frequent unrecognized words found in the FTB-3 corpus. We also created heuristic components to handle unrecognized proper names and abbreviations together with their inflected forms, giving them an appropriate morphological analysis. Finally, a transformation component was implemented to expand common contracted forms such as *jollei (if_not)*. An illustrative English example of this component would be the expansion of *don't* to *do not*. After the application of these techniques, only 2.2% of all tokens were left without a morphological analysis.

4.2 Dependency parsing

Dependency parsing of the morphologically tagged input is carried out using the *Mate-Tools*[4] parser of Bohnet (2010), a state-of-the-art graph-based statistical dependency parser. The parser is trained on the entire Turku Dependency Treebank transformed to the FTB scheme, as described in Section 3. The Mate-Tools parser was selected after a careful parsing accuracy comparison with the transition-based MaltParser of Nivre et al. (2007).

An issue particularly apparent in the legal text of the JRC-Acquis corpus is the $O(n^2)$ complexity of the dependency parser which makes parsing of long sentences exceeding 100 tokens impractical. In a large parsebank even a small proportion of such sentences becomes an issue as, ultimately, every parallel parsing process will be stuck parsing a long sentence, impairing the whole pipeline. Adopting a practical solution to the problem, we automatically split each sentence longer than 120 tokens to approximately evenly sized sections of roughly 100 tokens or less, and parse these sections separately. Candidate points to split the sentence are, in order of preference, after a semicolon, between items of numbered lists and, finally, after a comma or a colon. To reconstruct a full parse tree from the sections, we connect them using the classifier discussed in Section 3.3, only this time for performance reasons we connect the islands sequentially from left to right and only insert dependencies between the roots of the islands. This step affects 0.37% of all sentences in the parsebank. The possibility of parsing the long sentences with a transition-based parser was considered, however, we decided on the abovementioned procedure as it allows the re-use of the classifier as well as simplifies the software distribution of the final parsing pipeline.

Upon initial feedback from the FIN-CLARIN representatives, we have also implemented a separate post-processing step to address the cases where the parser produces two subject dependents of a verb (5.0% of all subjects). This is not a syntactically possible structure, however the purely statistical parser has no hard constraint preventing its generation. Further, double subject is an easily noticeable parsing error which we thus were specifically required to

[4] `http://code.google.com/p/mate-tools/`

address. Again relying on the classifier, in all multiple subject cases we preserve the subject with the highest score and replace all other subject dependencies with the highest-scoring non-subject dependency type.

4.3 Corpus text and parsing speed

The corpus to be parsed consists of 44.1M tokens from the JRC-Acquis corpus and 32.2M tokens from the EuroParl corpus. Parsing the total 76.4M tokens in 4.37M sentences (for an average sentence length of over 17 tokens) required approximately 900 CPU hours, using 4-core CPUs. This corresponds to processing speed of roughly 0.7 seconds per sentence, or, 24 tokens per second. The processing was split to 850 batches and parallelized on a cluster computer, resulting in actual parsing time of approximately 10 hours.

5 Evaluation

Since the parsebank was developed in a contract research setting, a formal evaluation, fully independent of us, was carried out by FIN-CLARIN and compared against pre-agreed acceptance thresholds. We did not carry out a separate evaluation ourselves since we do not have at our disposal the necessary gold-standard reference trees in the FTB scheme, and since there was no need for such an evaluation, the FIN-CLARIN results being the sole acceptance criterion of the output. The results of the evaluation are reported by Voutilainen et al. (2012b) and summarized here in Table 2. The accuracy of morphological information is in the 97-98% range while the dependency type accuracy (proportion of tokens whose dependency type is correct) and dependency relation accuracy (proportion of tokens whose governor is correct, usually referred to as *unlabeled attachment score*) are both in the 88-90% range. The commonly used *labeled attachment score* (i.e. the proportion of tokens which have both their governor and dependency type correctly assigned) is, unfortunately, not reported by Voutilainen et al. Overall, we can conclude that the accuracy of the parsebank is high, and above the acceptance threshold which was set to 95% for morphology accuracy, 85% for dependency type accuracy, and 87% for dependency relation accuracy.

Metric	Accuracy
Lemma	98%
Morphological analysis	97%
Dependency type	89–90%
Dependency relation	88–89%

Table 2: Official external evaluation results of the parsebank. The dependency type and dependency relation accuracies are reported as two values by Voutilainen et al. (2012b), the first value is obtained by directly inspecting the parsebank, while the second is obtained by comparison with an independently annotated gold standard.

The accuracy of the dependency scheme transformation procedure can be estimated directly on the 1,992 grammar examples from FTB-1 which we have annotated in the SD scheme and used when developing the transformation rules. The first, rule-based step of the transformation results in incomplete trees which we evaluate in terms of precision and recall of individual dependencies. The post-processed transformation forms complete trees, and is thus evaluated in terms of dependency type accuracy, dependency relation accuracy, and labeled attachment score. The first step results in precision of 83.4% and recall of 80.7%. The final transformed output after the machine learning based postprocessing, and after removing the null tokens,

has dependency type accuracy of 91.1%, dependency relation accuracy of 90.1%, and labeled attachment score of 88.2%.

The FTB-1 grammar examples is the only data at our disposal which allows a direct evaluation of the transformation. However, the grammar examples considerably differ in style from the target legal and parliamentary domain and are not fully representative of the transformation of the whole treebank. As a further, indirect evaluation we thus compare the FTB scheme parsing results reported in Table 2 with parsing results in the SD scheme on the JRC-Acquis and EuroParl sentences which we have annotated as in-domain data, as described in Section 2. Taking a weighted average in the proportions of JRC-Acquis and EuroParl texts in FTB-3, the dependency type and dependency relation accuracies for a parser trained on the SD scheme can be estimated as 92.3% and 89.1%. These figures are closely comparable with the figures achieved after transformation, as reported in Table 2. This suggests a successful transformation as the overall performance of the parser has not deteriorated compared to that prior to transformation.

6 Conclusions

In this paper, we have introduced the methods and resources used to build a large-scale Finnish dependency parsebank. Having started from an external, non-negotiable specification of the text to parse and the target scheme, we have developed a parsing pipeline capable of processing the 76.4 million token corpus, resulting in a parsebank with highly accurate morphological and syntactic annotation. The accuracy of the morphological information in the parsebank is in the 97-98% range whereas the accuracy of the dependency types and relations is in the 88-90% range, both figures being the result of an independent evaluation carried out by FIN-CLARIN.

We have applied several techniques to modify the existing resources to conform to the specification of the parsebank. In particular, we have introduced a dependency scheme transformation procedure with hand-written rules followed by a machine-learning based post-processing component. This procedure was used to transform the Turku Dependency Treebank into the target dependency scheme, enabling us to train a statistical dependency parser on this treebank.

The resulting parsebank is made freely available by FIN-CLARIN at
`http://www.ling.helsinki.fi/kieliteknologia/tutkimus/treebank/`.

Acknowledgments

Computational resources for the dependency parsing step of the pipeline were provided by CSC — IT Center for Science, Espoo, Finland.

References

Bohnet, B. (2010). Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING'10*, pages 89–97.

de Marneffe, M.-C. and Manning, C. (2008a). Stanford typed dependencies manual. Technical report, Stanford University. Revised for Stanford Parser v. 2.0.4 in November 2012.

de Marneffe, M.-C. and Manning, C. (2008b). Stanford typed dependencies representation. In *Proceedings of COLING'08, Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8.

Hakulinen, A., Vilkuna, M., Korhonen, R., Koivisto, V., Heinonen, T.-R., and Alho, I. (2004). *Iso suomen kielioppi / Grammar of Finnish*. Suomalaisen kirjallisuuden seura.

Haverinen, K. (2012). Syntax annotation guidelines for the Turku Dependency Treebank. Technical Report 1034, Turku Centre for Computer Science.

Haverinen, K., Ginter, F., Laippala, V., Kohonen, S., Viljanen, T., Nyblom, J., and Salakoski, T. (2011). A dependency-based analysis of treebank annotation errors. In *Proceedings of Depling'11*, pages 115–124.

Haverinen, K., Viljanen, T., Laippala, V., Kohonen, S., Ginter, F., and Salakoski, T. (2010). Treebanking Finnish. In *Proceedings of TLT9*, pages 79–90.

Koehn, P. (2005). Europarl: a parallel corpus for statistical machine translation. In *Proceedings of MT Summit X*, pages 79–86.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., and Marsi, E. (2007). MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135.

Pahikkala, T., Tsivtsivadze, E., Airola, A., Boberg, J., and Salakoski, T. (2007). Learning to rank with pairwise regularized least-squares. In Joachims, T., Li, H., Liu, T.-Y., and Zhai, C., editors, *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, pages 27–33.

Steinberger, R., Pouliquen, B., Widiger, A., Ignat, C., Erjavec, T., Tufiş, D., and Varga, D. (2006). The JRC-Acquis: A multilingual aligned parallel corpus with 20+ languages. In *Proceedings of LREC'06*, pages 2142–2147.

Voutilainen, A., Lindén, K., and Purtonen, T. (2011). Designing a dependency representation and grammar definition corpus for Finnish. In *Las tecnologías de la información y las comunicaciones: Presente y future en el análisis de córpora. Actas del III Congreso Internacional de Lingüística de Corpus*, pages 151–158.

Voutilainen, A., Purtonen, T., and Muhonen, K. (2012a). FinnTreeBank2 manual. Technical report, University of Helsinki, Department of Modern Languages.

Voutilainen, A., Purtonen, T., and Muhonen, K. (2012b). Outsourcing parsebanking: The FinnTreeBank project. In *Shall We Play the Festschrift Game?*, pages 117–132. Springer.

Published: 14 July 2013. Building the essential resources for Finnish: the Turku Dependency Treebank. Katri Haverinen. 1,2,4Â Due to their importance especially for statistical parsing, as well as many advanced applications, treebanks have been constructed for many languages, regardless of how widely spoken. Perhaps the best-known of the worldâ€™s treebanks are the Penn Treebank (Marcus et al. 1993) for English and the Prague Dependency Treebank (HajiÄ 1998) for Czech.Â For Finnish, the early versions of the Turku Dependency Treebank (TDT) constitute the first publicly available treebank (Haverinen et al. 2009 , 2010b , 2011). Attached tasks: DEPENDENCY PARSING. AddÂ Edit. Add Remove. DEPENDENCY PARSING. Results from the Paper. Edit. Add Remove. Submit results from this paper to get state-of-the-art GitHub badges and help the community compare results to other papers. Methods used in the Paper. Edit. In linguistics, a treebank is a parsed text corpus that annotates syntactic or semantic sentence structure. The construction of parsed corpora in the early 1990s revolutionized computational linguistics, which benefitted from large-scale empirical data. The exploitation of treebank data has been important ever since the first large-scale treebank, The Penn Treebank, was published. However, although originating in computational linguistics, the value of treebanks is becoming more widely appreciated in The corpus was created using heuristic extraction techniques in conjunction with an SVM-based classifier to select likely sentence-level paraphrases from a large corpus of topicclustered news data. These pairs were then submitted to human judges, who confirmed that 67% were in fact semantically equivalent. In addition to describing the corpus itself, we explore a number of issues that arose in defining guidelines for the human raters. View Publication. Groups.