Lang Resources & Evaluation
DOI 10.1007/s10579-013-9244-1

PROJECT NOTE

Building the essential resources for Finnish: the Turku Dependency Treebank

Katri Haverinen · Jenna Nyblom · Timo Viljanen · Veronika Laippala · Samuel Kohonen · Anna Missilä · Stina Ojala · Tapio Salakoski · Filip Ginter

Abstract In this paper, we present the final version of a publicly available treebank of Finnish, the Turku Dependency Treebank. The treebank contains 204,399 tokens (15,126 sentences) from 10 different text sources and has been manually annotated in a Finnish-specific version of the well-known Stanford Dependency scheme. The morphological analyses of the treebank have been assigned using a novel machine learning method to disambiguate readings given by an existing tool. As the second main contribution, we present the first open source Finnish dependency parser, trained on the newly introduced treebank. The parser achieves a labeled attachment score of 81 %. The treebank data as well as the parsing pipeline are available under an open license at http://bionlp.utu.fi/.

Keywords Treebank · Finnish · Parsing · Morphology

1 Introduction

The need for manually annotated resources, and more specifically treebanks, is widely acknowledged within the field of computational linguistics. Due to their

K. Haverinen (✉)
Turku Centre for Computer Science, University of Turku, Joukahaisenkatu 3–5,
20014 Turun yliopisto, Turku, Finland
e-mail: kahave@utu.fi

K. Haverinen · J. Nyblom · T. Viljanen · S. Kohonen · A. Missilä · S. Ojala ·
T. Salakoski · F. Ginter
Department of Information Technology, University of Turku, Turku, Finland

V. Laippala
Department of French Studies, University of Turku, Turku, Finland

K. Haverinen
University of Turku Graduate School, University of Turku, Joukahaisenkatu 3–5,
20014 Turun yliopisto, Turku, Finland

Published online: 14 July 2013

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 world's treebanks are the Penn Treebank (Marcus et al. 1993) for English and the Prague Dependency Treebank (Hajič 1998) for Czech. Other languages with treebanks completed or under construction include, among others, Swedish, Estonian, Dutch, Japanese, French, German, Hungarian, and even dead languages such as Latin and Ancient Greek.

For Finnish, the early versions of the Turku Dependency Treebank (TDT) constitute the first publicly available treebank (Haverinen et al. 2009, 2010b, 2011). Shortly after the second intermediate release of TDT, FinnTreeBank, a grammar-definition treebank consisting of all example sentences from the reference book of Finnish grammar by Hakulinen et al. (2004) was made available and later extended to a total of 169,450 tokens (Voutilainen and Lindén 2011). The differences between FinnTreeBank and TDT will be discussed in greater detail in Sect. 9. Finally, there also exists a small-scale treebank of 15,335 tokens from the narrow domain of clinical narratives, containing PropBank-style argument annotation in addition to morphology and dependency syntax (Haverinen et al. 2010a).

As the first main contribution of this paper, we present the final version of the Turku Dependency Treebank, considerably extended in both size and annotation scope relative to its previously available subsets. The treebank consists of 204,399 tokens (15,126 sentences), thus being the largest Finnish treebank in existence and more than twice the size of the latest previously available version. The treebank has been manually annotated using the well-known Stanford Dependency scheme and, in addition to the base-syntactic trees, it also contains a second layer of annotation with *conjunct propagation* and additional dependencies as described in the original Stanford Dependency scheme, another novel contribution in this paper. Unlike the earlier versions, the current release also contains a morphological layer that is based on an open source morphological analyzer, disambiguated using a novel machine learning method that relies on the unambiguous tokens for its training data, and the syntactic annotation for the features. The treebank is available at no cost, under an open license.

Several tools exist for morphological and syntactic analysis of Finnish. FinTWOL and FinCG (Koskenniemi 1983; Karlsson 1990) by Lingsoft Inc. are a commercial morphological analyzer and constraint grammar-based disambiguator. The Kielikone parser, an early commercial parser of Valkonen et al. (1987) both resolves morphological ambiguity and assigns basic syntactic functions. Recently, two open source morphological analyzers, OMorFi (Pirinen 2008; Lindén et al. 2009) and Voikko,[1] have become available as well. Compared to OMorFi, Voikko has a more limited lexicon coverage and is primarily applied in open source Finnish spellcheckers. Both are pure morphological analyzers with no disambiguation component. Finally, Connexor Machinese Syntax, another commercial tool, is the only currently available Finnish full dependency parser.[2]

[1] http://voikko.sourceforge.net/.

[2] http://www.connexor.eu.

The unavailability of an open source Finnish dependency parser was among the most important practical motivations for building the Turku Dependency Treebank, which provides the necessary data for statistical parser training. As the second main contribution of this paper, we therefore present a statistical parsing pipeline, consisting of a sentence splitter, a tokenizer, a morphological tagger and a full dependency parser. It constitutes the first freely available, open source dependency parser of Finnish, setting the initial baseline for Finnish statistical parsing.

In the following, we thoroughly discuss different aspects of the treebank as well as the accompanying tools, starting with the text selection criteria of the treebank in Sect. 2. We then move on to the syntactic annotation scheme in Sect. 3, and the annotation process in Sect. 4. The morphological analyses in the treebank are discussed in Sect. 5. Section 6 evaluates the quality of the syntax annotation as well as the morphological analyses. Section 8 describes the freely available parsing pipeline created using the treebank as well as briefly describes existing applications, while Sect. 9 discusses the differences between the Turku Dependency Treebank and FinnTreeBank. Sect. 10 concludes the work.

2 Text selection

The Turku Dependency Treebank consists of 204,399 tokens (15,126 sentences) of written Finnish, from ten different text sources or genres. 10 % of this data, as calculated on the level of documents, is held out as a test set, for the purpose of parser comparisons and scientific challenges. This test set is available to the public via a web-based parser evaluation service, which is described below in Sect. 7 Unless specifically otherwise stated, all numbers presented in this paper, in this as well as all of the following sections, are calculated on the full treebank, that is, including also the test set.

When selecting the text for the treebank, we have used two broad criteria. First of all, the treebank is to be made publicly available under a license that does not restrict its use. Therefore, the selected text should either be released under an open license originally, or it should be possible, with reasonable effort, to negotiate such a license. The specific license used for the treebank is the *Creative Commons Attribution-Share Alike license*, which allows both non-commercial and commercial use.

In addition, the treebank should exhibit topical variety. This criterion is exercised in two ways. First, the treebank consists of 10 *sections*, each of which contains text from a different source. Each section consists of a number of *documents*, which are single, continuous texts—with the exception of the grammar example section, as described below. The different sections and their sentence and token counts, as well as the numbers of documents included, are listed in Table 1. In order to avoid biasing the treebank towards the topics of long articles,[3] documents that are at most 75 sentences long have been included in the treebank and annotated in their entirety, and documents longer than that have been truncated at 75 sentences.

[3] For instance, the Finnish Wikipedia contains articles that are more than 300 sentences long.

Table 1 Sections of TDT and their sizes in terms of document, sentence and token counts

Section	Documents	Sentences	Tokens
Wikipedia articles	200	2,269	32,272
Wikinews articles	100	1,120	14,497
University online news	50	942	13,283
Blog entries	77	1,781	22,403
Student magazine articles	23	1,058	14,432
Grammar examples	–	1,992	17,061
Europarl speeches	80	1,082	19,964
JRC-Acquis legislation	29	1,141	24,909
Financial news	50	1,002	12,689
Fiction	65	2,739	32,889
All	674	15,126	204,399

No document count is given for the grammar examples section, as this section consists of individual sentences with no further structure

Second, we have strived for topical variety within each section. In the following, we discuss the section-specific methods for selecting the documents in a manner that ensures variety in that section. For all text sources, we have either used random selection or selected texts in order from the newest to the oldest. Several of the sources were published under an open license to begin with, but for those which were not, we have contacted the authors individually to gain permission to re-publish their text.

As the Finnish Wikipedia contains a large number of articles by different authors and concerning a variety of topics, the articles in the Wikipedia section of the treebank have been selected randomly. In the Finnish Wikinews, the number of different authors is smaller, and therefore we have first randomly selected an author and then randomly selected an article by them.

The university news section has been gathered from the UtuOnline magazine, which is an online magazine of the University of Turku. These texts were selected in order starting from the newest articles. The size of the section was restricted to 50 articles, as the topics are fairly limited.

The blog entries used in TDT were collected from top ranking items on various lists of popular blogs. A rough topic-wise division was made, and the number of blogs to be selected on each topic was limited. The blogs used in the treebank represent the following topic categories: *personal and general*, *style and fashion*, *relationships and sex*, *technology*, *living abroad* and *cooking*. From each blog, approximately 200 sentences of text were used, selecting entries from the newest to the oldest. Potentially problematic entries, such as those containing long quotes that could cause copyright issues, were disregarded in the selection process.

The student magazine texts were selected from magazines of three different student organizations: one magazine for computer science students, one for mathematics and physics students, and one for social sciences students. These magazines are written for students by students, and they range over a variety of topics and styles, from writings about studies and student life to letters to the editor and humorous texts. The newest electronically available issues of the three magazines were used, and from those, at most two texts by the same author were selected, as student magazines are often produced by a small number of active writers.

The grammar examples section consists of example sentences and fragments from the Finnish Grammar (Hakulinen et al. 2004), a random subset of FinnTreeBank. As will be discussed in Sect. 9, this section of the treebank allows for a comparison of the treebanks, as well as a conversion between their syntactic schemes. Unlike the sentences in other sections of TDT, the grammar examples do not form a continuous story. The original FinnTreeBank contains some duplicate sentences, as the grammar on occasion uses the same examples for multiple phenomena. These duplicates were removed from TDT.

Two sections of the treebank are based on existing corpora mostly aimed at machine translation; one section consists of speeches from the Finnish section of the Europarl corpus (Koehn 2005), and one of legislation text from the JRC-Acquis corpus (Steinberger et al. 2006). Random selection was used for both of these text sources. From Europarl, we selected random speech turns, randomizing also the meeting and topic from which the speech turns were selected. From JRC-Acquis, in turn, we selected random documents from random years.

The financial news section of the treebank uses the articles of a Finnish newspaper focusing on financial news, *Taloussanomat*. The majority of the texts of the newspaper are originally published under an open license, and only these texts were selected for the treebank, as we had the permission to re-license these specific texts to fit the license of the treebank.

The fiction section consists of texts by amateur writers from various sources on the web. Most commonly they are short stories posted in a dedicated blog. In the case of short stories, each story was considered a separate text, and at most 75 sentences from each text were included in the treebank, as usual. In the case of longer, continuous stories, such as serials, the whole story was considered a single text, and only the first 75 sentences from the beginning of the first chapter were selected for annotation.

3 Dependency annotation scheme

The annotation scheme of the treebank is a Finnish-specific version of the well-known Stanford Dependency (SD) scheme, originally developed by de Marneffe and Manning (2008a, b). The SD scheme represents the syntax of a sentence as a graph where the nodes represent the words of the sentence, and the edges represent directed dependencies between them. One of the two words connected by a dependency is the *head* or *governor* while the other is the *dependent*. Each dependency is labeled with a *dependency type*, which describes the syntactic function of the dependent word. Figure 1 illustrates the Stanford Dependency scheme on a Finnish sentence.

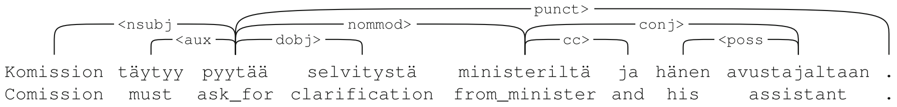

punct>
<nsubj
nommod>
conj>
<aux
dobj>
cc>
<poss
Komission täytyy pyytää selvitystä ministeriltä ja hänen avustajaltaan .
Comission must ask_for clarification from_minister and his assistant .

Fig. 1 The stanford dependency scheme. The sentence can be translated as *The commission must ask for clarification from the minister and his assistant*

SD is becoming a popular choice of syntax scheme in multiple languages. Treebanks natively annotated in SD include a treebank for Chinese (Lee and Kong 2012) and one for Persian (Seraji et al. 2012). With the conversion included in the original Stanford tools,[4] the Penn Treebank (Marcus et al. 1993) and indeed any treebank annotated in the Penn Treebank constituency scheme can be converted into the SD scheme. In addition, the SD scheme is especially popular in parser evaluation works (Cer et al. 2010; Nivre et al. 2010; Clegg and Shepherd 2007; Miwa et al. 2010; Foster et al. 2011), and several parsers are capable of producing the scheme either natively or by conversion, including the Charniak-Johnson parser (Charniak and Johnson 2005), the Stanford parser (Klein and Manning 2003), the Clear parser Choi and Palmer (2011), the parser of Tratz and Hovy (2011), and naturally any dependency parser that can be trained from a treebank, such as the MaltParser (Nivre et al. 2007), the MSTParser (McDonald et al. 2006) or the Mate-Tools parser (Bohnet 2010). The scheme was originally intended to be application-oriented, and it has indeed been successfully used in applications, particularly in the biomedical domain (Björne et al. 2010; Miyao et al. 2009; Qian and Zhou 2012), and otherwise in opinion extraction (Zhuang et al. 2006) and sentiment analysis (Meena and Prabhakar 2007).

The original SD scheme of de Marneffe and Manning has four *variants*, which include a different subset of dependency types each, describing different levels of syntactic and semantic detail. The annotation in the Turku Dependency Treebank consists of two different layers. The first layer is based on the *basic* variant of SD. The analyses are trees, with the exception of one dependency type concerning multi-word named entities, which is allowed to break the tree structure. However, we also provide a strict tree version of the treebank, as will be described in Sect. 7. The annotation in the first layer is described in Sect. 3.1. The second layer of annotation adds additional dependencies on top of the first layer; this results in analyses that are no longer trees, but rather directed graphs. The second layer of annotation is discussed in Sect. 3.2.

The dependency types of the original SD scheme are arranged in a hierarchy, where the most general dependency type *dep* is on top of the hierarchy, and all other types are its direct or indirect subtypes. When annotating using SD, the most specific dependency type possible is always to be selected from the hierarchy. The scheme defines a total of 55 dependency types, including six types which are intermediate in the hierarchy and rarely used. The remaining 49 types include 48 bottom level types and the most general type *dep*.

[4] http://nlp.stanford.edu/software/lex-parser.shtml

The Finnish-specific version of the scheme has been modified from the original scheme by removing dependency types where the corresponding phenomenon does not occur in Finnish, and adding new types where a phenomenon has not been covered by the original scheme. The resulting scheme used in TDT has in total 53 dependency types, including 46 types in the first, syntactic layer, four intermediate types that are present in the (modified) SD hierarchy but not used in the TDT annotation, and three types that are only present in the second layer of annotation discussed in Sect. 3.2. All these types are listed in Table 2. The annotation scheme has been described in detail in the TDT annotation manual (Haverinen 2012); in this work we focus on the differences between the Finnish and English schemes.

3.1 The finnish-specific SD scheme: the first annotation layer

Some phenomena of the Finnish language required modifications to the scheme, and some more general features were unaccounted for in the original SD scheme, but overall the modifications made were small-scale, so as to remain consistent with other SD-based resources. These changes are discussed in the following two subsections.

3.1.1 Additions to the SD scheme

Perhaps the most notable difference between the original SD scheme and the Finnish-specific version concerns nominal modifiers and adpositions. The Finnish language includes both pre- and postpositions, but inflected nominal modifiers without an adposition are often used instead. Sometimes the very same meaning can be expressed in both of these ways, and semantically, nominal modifiers and adpositional phrases are similar. In order to analyze them similarly also on the level of syntax, we have introduced a new type for inflected nominal modifiers, *nommod*. This type has two uses: it can be used alone for an inflected nominal modifier without an adposition, or it can be combined with a second new type, *adpos* (adposition). Unlike in the English SD scheme, the nominal is considered the head, again in order to analyze nominal modifiers and adpositional phrases similarly. For an illustration of nominal modifiers and adpositional phrases, see Fig. 2.

Next, a Finnish genitive modifier may convey many different meanings. Most of these are not distinguished in TDT, but we have added two new dependency types for an important and frequent phenomenon: genitive subjects (*gsubj*) and objects (*gobj*) of a noun. For an illustration of these two new types, see Fig. 3.

Another Finnish-specific dependency type added to the scheme relates to expressions of owning and having. In Finnish, clauses that express owning, *omistuslause (possessive clause)* (Hakulinen et al. 2004, §891), are somewhat different from their English counterparts, as there is no separate verb with the meaning *to have*, but rather the verb *olla* (*to be*) is used instead. For instance, the sentence *I have a cat* would be expressed as *Minulla on kissa*, which in turn could be literally translated as *“At me is a cat”*. The Finnish possessive clauses resemble another clause type, namely *existential clauses*, such as *Pihalla on kissa* (*There is a*

Table 2 Dependency types of the Finnish-specific SD scheme

Dependency type	Description
dep	dependent (most general dependency type)
aux	auxiliary
auxpass	passive auxiliary
cop	copula
neg	negation marker
arg	argument
comp	complement
acomp	adjectival complement
ccomp	clausal complement
iccomp	infinite clausal complement
complm	complementizer
dobj	direct object
mark	marker (subordinating conjunction in adverbial clauses)
rel	relativizer, relative word
xcomp	open clausal complement (complement shares main verb's subject)
subj	subject
csubj	clausal subject
csubj-cop	clausal copular subject
nsubj	nominal subject
nsubj-cop	nominal copular subject
xsubj*	external subject
xsubj-cop*	external copular subject
cc	coordinating conjunction
compar	comparative
comparator	comparative conjunction
conj	coordination
ellipsis*	ellipsis of a head word
intj	interjection
mod	modifier
adpos	adposition
advcl	adverbial clause modifier
advmod	adverb modifier
amod	adjectival modifier
appos	apposition
det	determiner
infmod	infinitive modifier
nn	noun compound modifier or appellation modifier
nommod	nominal modifier
nommod-own	nominal modifier signifying the owner in a possessive clause
num	numeral modifier
number	numerical expression
partmod	participal modifier
poss	genitive modifier
gobj	genitive object (of a noun)
gsubj	genitive subject (of a noun)
preconj	preconjunction
prt	phrasal particle
quantmod	quantification modifier
rcmod	relative clause modifier
name	multi-word named entity
parataxis	parataxis
punct	punctuation
voc	vocative

Types new to this scheme version are emphasized, and types only present in the second layer of annotation are marked with an asterisk

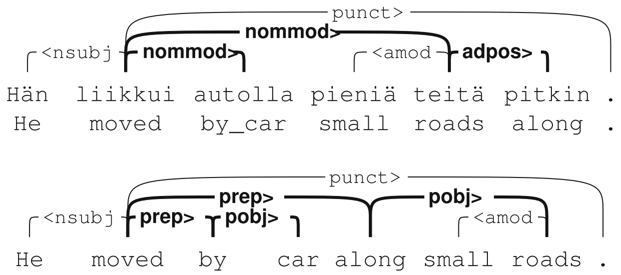

punct>
nommod>
<nsubj
nommod>
<amod
adpos>
Hän liikkui autolla pieniä teitä pitkin .
He moved by_car small roads along .
punct>
prep>
pobj>
<nsubj
prep>
pobj>
<amod
He moved by car along small roads .

Fig. 2 Nominal modifiers and adpositions in Finnish (*top*). Note how in adpositional phrases, the nominal is made the governor of the phrase. The sentence can be translated as *He moved by car along small roads*. For comparison, also the corresponding English sentence and its analysis in the original SD scheme is given (*bottom*)

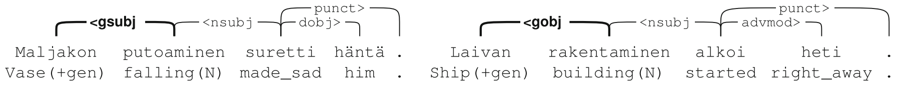

<gsubj
<nsubj
punct>
dobj>
Maljakon putoaminen suretti häntä .
Vase(+gen) falling(N) made_sad him .
<gobj
<nsubj
punct>
advmod>
Laivan rakentaminen alkoi heti .
Ship(+gen) building(N) started right_away .

Fig. 3 Genitive subjects (*left*) and objects (*right*) of a noun. The examples can be translated as *The falling of the vase made him sad* and *The building of the ship started right away*

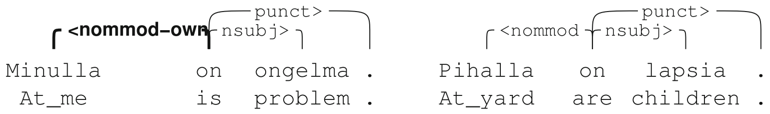


Fig. 4 A possessive clause (omistuslause, *left*) as compared to an existential clause (*right*). Otherwise the analysis is exactly the same, but in the possessive clause the owner is marked using the dedicated dependency type *nommod-own*. The examples can be translated as *I have a problem* for the possessive clause, and *There are children in the yard* for the existential clause

cat in the yard). In fact, Hakulinen et al. (2004, §891) consider possessive clauses a subtype of existential clauses. Theories differ in whether they consider the nominative/partitive sentence element in existential or possessive clauses to be a subject or not; for instance Helasvuo and Huumo (2010) argue that this sentence element is not in fact a subject and term it *e-NP* instead, whereas Hakulinen et al. (2004, §910) consider the *e-subject* simply "the subject of an existential clause".

The possessive clauses in TDT are analyzed similarly to existential clauses. In both of these clause types, the element corresponding to the e-subject (*kissa*, *cat*) is marked as the subject, and the adessive sentence element (*minulla*, *"at me"*) as a nominal modifier. As the possessive clause is clearly a very specific clause type with its own meaning, these structures are marked in TDT with the separate dependency type *nommod-own*, which is a subtype of nominal modifier, *nommod*. As this analysis is consistent, it is possible to transform it according to any view desired. Figure 4 shows an example of the TDT analysis of a possessive clause, and as a point of comparison, the analysis of an existential clause.

A few more general additions to the SD scheme were also required for the annotation of TDT. Most importantly, a solution was needed for situations where the head word of a phrase is absent from the text, but its dependents are present. This would be problematic for any dependency scheme, as the head word is needed in order to construct an analysis. There are two different cases where a missing head word can occur, and both are treated similarly in TDT. First, the head word of a clause can be missing in *fragments*, which are common in for instance newspaper titles. An example of such a sentence would be *Presidentti Kiinaan solmimaan sopimusta (The president to China to make a contract)*. Second, a head word may be missing in *gapping*, a type of *ellipsis* where the head word of a phrase is omitted to avoid repetition while its dependents are present. For example, the sentence *Maija luki kirjaa ja Matti sanomalehteä (Maija read a book and Matti a newspaper)* contains a gapping structure.

In TDT, when a word is absent from a sentence and it is necessary in order to be able to construct an analysis, a *null token*, which represents the missing word, is inserted during annotation. Similar solutions to this issue have been adopted in several other dependency treebanks, for instance the dependency version of the TIGER treebank for German (Brants et al. 2002), the SynTagRus treebank of Russian (Boguslavsky et al. 2002), the Hungarian Dependency Treebank (Vincze et al. 2010) as well as the Hindi treebank of Begum et al. (2008) and the Arabic treebank of Dukes and Buckwalter (2010).

Null tokens are only inserted in TDT when they are needed as the governor of another token. Thus, not all forms of ellipsis are marked by null tokens, nor is a null token inserted for omitted copulas or auxiliaries. This is because in the SD scheme, the head of a copular clause is the predicative, not the copular verb, and additionally, if a copula or an auxiliary is absent, its dependents are also absent. The majority of the null tokens (651/706) are verbs, but also other parts-of-speech are possible, mainly in gapping situations. See Fig. 5 for an illustration of both uses of the null token.

The Finnish-specific SD scheme also accounts for multi-word named entities, which are marked using the dependency type *name*. This dependency type is

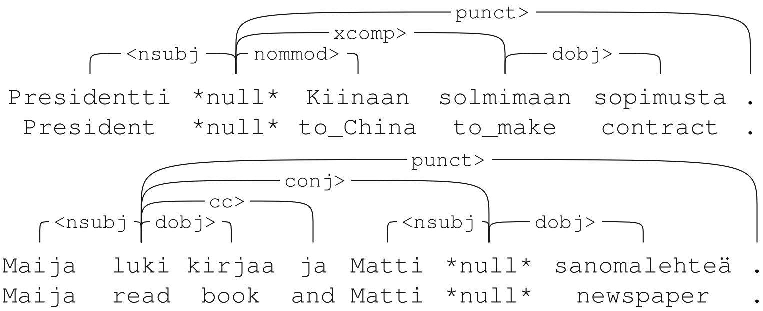

punct>
xcomp>
<nsubj
nommod>
dobj>
Presidentti *null* Kiinaan solmimaan sopimusta .
President *null* to_China to_make contract .
punct>
conj>
cc>
<nsubj
dobj>
<nsubj
dobj>
Maija luki kirjaa ja Matti *null* sanomalehteä .
Maija read book and Matti *null* newspaper .

Fig. 5 Fragments (*top*) and gapping (*bottom*). When a word necessary for constructing an analysis is missing, a *null token* is inserted to represent it. The fragment example can be translated as *The president to China to make a contract* and the gapping example as *Maija read a book and Matti a newspaper*

exceptional in the sense that it is allowed to break the tree structure. However, the analyses can be processed so as to make all sentence structures trees, as will be discussed in Sect. 7 The governor of a *name* dependency is the rightmost word of the named entity, and the dependent the leftmost. If there are more than two words in the entity, no additional *name* dependencies are marked. However, if the named entity has an obvious internal syntactic structure, this structure is marked in addition to the *name* dependency. In these cases, the head word of the named entity is the actual head, not the rightmost token. Figure 6 illustrates both usages of the *name* dependency type. Note that the analysis of the internal structure of a named entity can also be partial, if the entity consists of different parts, where some parts have an internal structure and some do not. The rationale behind the twofold analysis of named entities is that we wish to allow the user to easily transform and re-interpret the annotation as desired and to limit future applications as little as possible.

As smaller modifications, we have added to the Finnish-specific scheme dependency types for vocatives (*voc*) and interjections (*intj*), which where previously unaccounted for in SD. For comparative structures, we have introduced two types, *compar* and *comparator*, where the former connects the comparative word and the object of comparison, and the latter marks the comparative conjunction, if present. In addition, we have introduced separate types for subjects of copular clauses, as these clauses have their own special treatment in SD. This adds two new types: *nsubj-cop* for nominal subjects and *csubj-cop* for clausal subjects. Finally, we add the type *iccomp* for infinite clausal complements.

In the second layer of annotation that will be discussed in Sect. 3.2, we have added a separate type for external copular subjects, *xsubj-cop*, analogously to the type *nsubj-cop* in the base-syntactic layer. Also the dependency type *ellipsis* marking gapping structures is new to the second annotation layer.

3.1.2 Removals from the SD Scheme

Some phenomena of the English language accounted for in the SD scheme do not occur in Finnish, rendering the corresponding types unnecessary. These types have been removed from the Finnish-specific SD scheme. Passive clauses do not have subjects in Finnish (see for instance the Finnish grammar by Hakulinen et al. (2004, §1313)), and consequently, the passive subject types (*nsubjpass* and *csubjpass*) from the English scheme version are not used in TDT. What in English is considered the passive subject, is in Finnish the direct object, and thus the type *dobj* is used instead. The *agent* type, intended for agents of passive clauses, is similarly not needed for

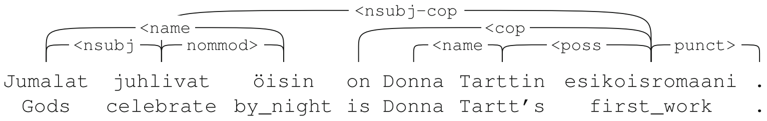


Fig. 6 Multi-word named entities are marked with the dependency type *name*. Note how *Jumalat juhlivat öisin* has an internal structure (*"Gods celebrate by night"*, the Finnish title of *A secret history*), where the main verb, *juhlivat*, acts as the governor, whereas *Donna Tartt* is merely a name. The sentence can be translated as *A secret history is Donna Tartt's first work*

Finnish, as there is no agent construction for passives. In addition, we consider the type *agent* semantic rather than syntactic. Certain constructions, such as *toimesta* and *taholta* (see the Finnish grammar (Hakulinen et al. 2004, §1327)), however resemble the English agent structure. They are analyzed as nominal modifiers, in accordance with the commonly used Finnish morphological analyzers, FinTWOL and OMorFi. Other removed types include types for the expletive *there* (*expl*), the indirect object (*iobj*), and the possessive *'s* (*possessive*), none of which occur in Finnish. As discussed above, adpositional phrases are treated differently from the original SD scheme, meaning that also the preposition-related types from the original scheme, *prep* and *pobj*, have been removed. At this point in time, *referents* in relative clauses (*ref*) are not annotated in TDT. When used, this type violates the treeness condition, and therefore it would belong to an additional layer of annotation.

Three types from the original SD scheme, *purpcl* (purpose clause), *tmod* (temporal modifier) and *measure* were considered semantic in nature and were not used in the syntax annotation, but rather the appropriate syntactic types were used. Additionally, the original SD scheme contains a type for apposition-like abbreviations (*abbrev*), used in contexts such as *National Aeronautics and Space Administration (NASA)*. In TDT, only the more general type for appositions (*appos*) is used since abbreviations are identified in the morphological analysis. Finally, predicatives are always analyzed as predicatives, rather than attributives (*attr*) as is possible in the original SD scheme.

3.2 The second annotation layer: conjunct propagation and extra dependencies

The annotation in the second layer of TDT covers the following phenomena: propagation of conjunct dependencies, external subjects, syntactic functions of relativizers, and gapping. In the following, each of these four phenomena are discussed in turn.

Conjunct propagation The first and most important phenomenon covered in the second annotation layer of TDT is *propagation of conjunct dependencies*, as it is called by de Marneffe and Manning (2008a). This phenomenon concerns coordination structures. In the SD scheme, the first coordinated element is the head of the coordination, and the rest of the coordinated elements as well as the coordinating conjunction depend on it. If a sentence element modifies the head of a coordination, it may be that it in fact modifies all or some of the coordinated elements and should therefore be *propagated* to them. Similarly, if the head of a coordination modifies another sentence element, it is possible that all or some of the coordination members act as the modifiers. For an illustration of a sentence annotated with the conjunct propagation, see Fig. 7.

In addition to merely propagating to other coordinated elements, it is possible for a dependency to simultaneously change its type. This can occur for instance if the elements coordinated are of different parts-of-speech, or if the same sentence element plays a different role to a second predicate. Figure 8 illustrates conjunct propagation with dependency type changes.

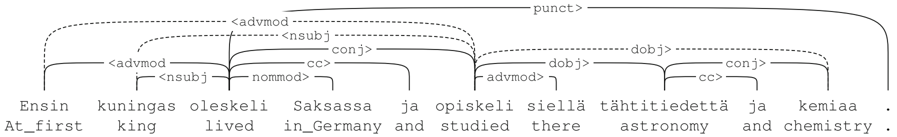

punct>
<advmod
<nsubj
conj>
dobj>
<advmod
cc>
dobj>
conj>
<nsubj
nommod>
advmod>
cc>
Ensin kuningas oleskeli Saksassa ja opiskeli siellä tähtitiedettä ja kemiaa .
At_first king lived in_Germany and studied there astronomy and chemistry .

Fig. 7 Propagation of conjunct dependencies. The base layer of annotation is marked with solid dependencies, and the propagated dependencies are dashed. The example sentence can be translated as *First the King lived in Germany and studied there astronomy and chemistry*. Note how the subject (*kuningas*, *king*) and the adverb modifier (*ensin*, *first*) propagate from the first verb to the second, but the nominal modifier (*Saksassa*, *in Germany*) does not. Also note how the direct object dependency arriving to the second coordination propagates

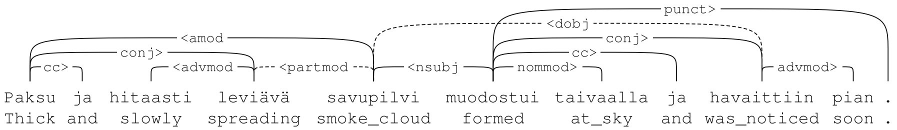

punct>
<dobj
<amod
conj>
conj>
cc>
cc>
<advmod
<partmod
<nsubj
nommod>
advmod>
Paksu ja hitaasti leviävä savupilvi muodostui taivaalla ja havaittiin pian .
Thick and slowly spreading smoke_cloud formed at_sky and was_noticed soon .

Fig. 8 Propagation of conjunct dependencies with dependency type changes. On the *left*, the adjectival modifier is coordinated with a participal modifier, and thus the type of the *amod* dependency changes into *partmod* while propagating. On the *right*, the word *savupilvi* (*cloud of smoke*) acts as the subject of the first clause, but as the object of the second clause, and hence the type of the propagated dependency changes. The example can be translated as *A thick and slowly spreading cloud of smoke formed at the sky and was soon noticed*

The existing Stanford tools[5] are able to produce output with the propagated dependencies present; however, de Marneffe and Manning (2008a) note that this part of the tools performs imperfectly. To our knowledge, TDT is the first existing treebank with manually annotated conjunct propagation in the SD scheme.

External subjects The second phenomenon annotated in the second layer of TDT is *external subjects*, marked with the dependency type *xsubj* (or *xsubj-cop*, for copular external subjects). With open clausal complements, the main verb and the clausal complement share a subject (subject control). The fact that the subject of the first verb also acts as the subject of the second verb cannot be marked in the base layer of annotation due to the treeness restriction, and hence these dependencies are only marked in the second layer. It should be noted that external subjects interact with the conjunct propagation both ways: external subjects can propagate, and also propagated subjects can produce an external subject. Figure 9 serves as an illustration of external subjects.

Syntactic functions of relativizers The third phenomenon annotated in the second layer concerns relative clauses. In the base syntactic layer, the phrase containing the relative word is marked simply as a *relativizer*, *rel*. However, the relativizer also always has a secondary syntactic function. For instance, the word *joka* (*which*) can act as the subject of the relative clause. In the base layer of annotation, this information is omitted, again due to the treeness restriction. Thus, in the second layer, each relativizer is given its syntactic function by adding a new dependency

[5] http://nlp.stanford.edu/software/lex-parser.shtml

that corresponds to the existing *relativizer* dependency in the first layer. The two dependencies usually coincide with respect to their head and dependent words, but as the governor of a relativizer dependency is always the main predicate of the relative clause, this is not always the case. The type of the second-layer dependency is one of the 46 dependency types defined in the first layer of the scheme. For an illustration, see Fig. 10.

Similarly to external subjects, also relativizers can propagate in coordinations. In addition, if a relativizer acts as a subject to a verb, it can also act as an external subject to an open clausal complement of this verb.

Gapping Language contains several different types of ellipsis, but only one of them is explicitly marked in TDT, namely the omission of a governor, *gapping*. Gapping is marked with null tokens (see Sect. 3.1) as well as a semantic dependency of the type *ellipsis*. See Fig. 11 for an illustration. In addition to gapping, some elliptical phenomena are marked less explicitly as propagated dependencies.

3.3 Discussion

One of the design-principles of the SD scheme, as originally created by de Marneffe and Manning (2008b), was language independence. From this perspective, the revisions required for Finnish were small-scale in general, and the overall scheme appears to be suitable for Finnish. Some of the revisions made for Finnish are also

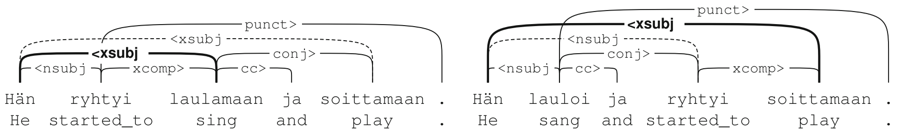

punct>
<xsubj
<xsubj
conj>
<nsubj
xcomp>
cc>
Hän ryhtyi laulamaan ja soittamaan .
He started_to sing and play .
punct>
<xsubj
<nsubj
conj>
<nsubj
cc>
xcomp>
Hän lauloi ja ryhtyi soittamaan .
He sang and started_to play .

Fig. 9 External subjects. Note how the *xsubj* dependencies can propagate (*left*), and how propagated subjects can produce an external subject (*right*). The examples can be translated as *He started to sing and play* and *He sang and started to play*

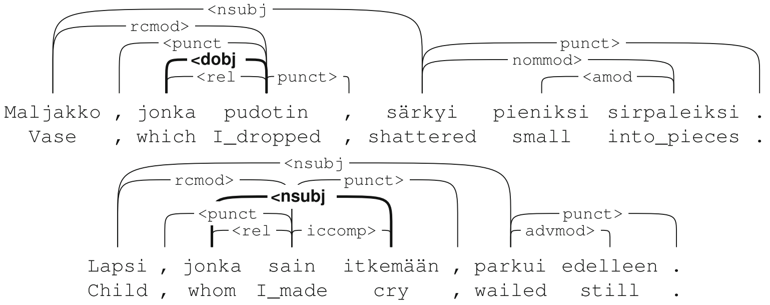

<nsubj
rcmod>
<punct
punct>
nommod>
<dobj
<amod
<rel
punct>
Maljakko , jonka pudotin , särkyi pieniksi sirpaleiksi .
Vase , which I_dropped , shattered small into_pieces .
<nsubj
rcmod>
punct>
<nsubj
<punct
punct>
<rel
iccomp>
advmod>
Lapsi , jonka sain itkemään , parkui edelleen .
Child , whom I_made cry , wailed still .

Fig. 10 Syntactic functions of relativizers. A relativizer can act in any syntactic function, such as an object (*top*) or a subject (*bottom*). Note that in the bottom-most example, the *nsubj* dependency does not coincide with the *rel* dependency that it corresponds to on the first layer. The examples can be translated as *The vase that I dropped shattered into small pieces* and *The child whom I made cry still wailed*

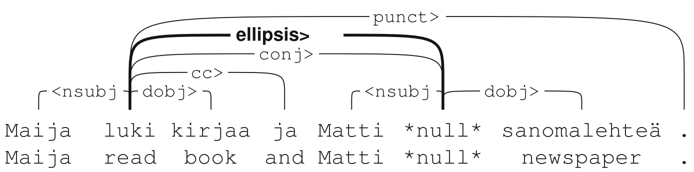

punct>
ellipsis>
conj>
cc>
<nsubj
dobj>
<nsubj
dobj>
Maija luki kirjaa ja Matti *null* sanomalehteä .
Maija read book and Matti *null* newspaper .

Fig. 11 Gapping is marked by a null token to represent the elided word, as well as a dependency of the type *ellipsis*. The example can be translated as *Maija read a book and Matti a newspaper*

more generally applicable and should be considered in future SD scheme annotation efforts.

Perhaps the most notable of these general revisions is the treatment of adpositions. The preposition-as-head analysis is suitable for English, but for a language that expresses the same meanings using either the case system or adpositions, such an analysis seems inconsistent. Almost regardless of language, some solution is also required for fragmentary and elliptical phenomena, which we have addressed using additional *null tokens*. Smaller issues likely to be encountered also in languages other than Finnish and English include vocatives, interjections, comparative structures and multi-word named entities, which have no predefined analysis in the original SD scheme. For languages that lack a separate verb for *having*, a special analysis that distinguishes possessive and existential clauses is called for. Marking copular subjects using the *-cop* types may be beneficial for a number of languages and genres, as it allows easy identification of copular clauses even in cases where the copular verb is absent. Finally, depending on the desired granularity of the scheme, genitive modifiers could be classified into types other than the possessive type, which is due to the roots of the scheme being in the English language.

In general, if the addition of a new type is desired for a specific language, the type hierarchy of the SD scheme is of assistance. If new types are inserted in a suitable place in the hierarchy, they can easily be replaced by their supertypes in applications requiring a more coarse-grained analysis, or comparability with other corpora annotated in the SD scheme.

4 Annotation process

In the course of the annotation process, there have been in total seven different annotators contributing to the treebank, with varying backgrounds and different amounts of previous experience. Out of these seven annotators, five have contributed to the first annotation layer, and six to the second.

The first and second layer of annotation described in Sect. 3 were annotated in two consequent steps, so that the second annotation layer was based on the existing first layer. For both layers, we used a custom annotation tool that is able to show the analyses visually, and an early version of this tool is publicly available on the treebank website. We begin this Section by describing the general workflow of the annotation, which applies to both the first and second layers, and continue by describing the specifics of the two layers.

4.1 Annotation workflow

The annotation protocol of the treebank in its entirety is *full double-annotation*. The annotation process consists of three phases, which result in three different kinds of annotations.

Individual annotations Each document is first assigned to two different annotators, who annotate it independently of each other. This results in two *individual* annotations.

Merged annotation Next, the two *individual* annotations of the same document are automatically merged into a single analysis, so that both analyses are shown whenever there is a difference. These differences are then settled in a meeting of the whole annotation team. This results in a so called *merged* annotation of the document.

Final annotation After settling the differences, an additional phase of corrections is needed in order to gain the *final* annotation of the treebank, for two reasons. First, as the annotation team comes across new examples, some annotation decisions may change over time, and thus older annotations will become outdated. In order to make even old annotations conform to the newest annotation decisions, *consistency checks and corrections* are performed. Second, possible sentence splitting and tokenization issues are corrected at this stage, in order to produce high-quality annotations for the treebank while keeping the double-annotation and merging process as simple as possible. This procedure also has the additional benefit that it provides perfectly aligned data for studying the annotation process itself, using the *individual* and *merged* annotations.

4.2 The first annotation layer

The first layer of the treebank was annotated by pairs of annotators from a pool of five different annotators. Taking into account the constraints of the annotators available at each time and the proportion of time they could dedicate to the annotation, the documents were divided between annotators as equally as possible. Also, care was taken that all different possible pairs of annotators were given documents to annotate against each other, again taking into account the previously mentioned constraints and the additional requirement that in the beginning of a new annotator's training, the annotator must only annotate against the annotator-in-chief, Annotator 1, as this annotator was the most experienced and the most accurate, as will be shown in Sect. 6.1. This was to make sure that as many of the beginning annotator's mistakes as possible are eliminated in the double-annotation. The contributions of each annotator in each section are presented in Table 3.

A substantial part of the first layer (10,863 sentences, 146,790 tokens) has been annotated using a parser as an aid. That is, after an initial phase of annotating all sentences from scratch, we have used the completed part of the treebank to train a

Table 3 Annotator contributions per section in the base syntax layer

	Ann. 1		Ann. 2		Ann. 3		Ann. 4		Ann. 5	
	Tokens	%	Tokens	%	Tokens	%	Tokens	%	Tokens	%
Wikipedia	25,424	39.4	19,215	29.8	16,928	26.2	0	0	2,977	4.6
Wikinews	13,295	45.9	13,483	46.5	1,202	4.1	1,014	3.5	0	0
Uni. news	9,380	35.3	6,111	23.0	5,504	20.7	5,571	21.0	0	0
Blogs	17,792	39.7	18,464	41.2	0	0	0	0	8,550	19.1
Student	10,656	36.9	8,147	28.2	6,692	23.2	3,369	11.7	0	0
Grammar	12,169	35.7	9,545	28.0	2,163	6.3	0	0	10,245	30.0
Europarl	16,898	42.3	14,022	35.1	5,094	12.8	797	2.0	3,117	7.8
JRC-Acquis	16,438	33.0	13,977	28.1	2,917	5.9	0	0	16,486	33.1
Financial	9,061	35.7	9,054	35.7	0	0	0	0	7,263	28.6
Fiction	25,296	38.5	20,739	31.5	0	0	0	0	19,743	30.0
Overall	156,409	38.3	132,757	32.5	40,500	9.9	10,751	2.6	68,381	16.7

For each annotator is given the amount of tokens annotated in each section (tokens), as well as the percentage of the section annotated (%). The overall amount of tokens annotated in each section (100 %) is twice the size of the section, as each token was counted twice, once for each annotator. Thus the theoretical maximum of tokens that one annotator could annotate is 50 % of the total, seeing that each document must have two annotators

statistical parser, using the MaltParser system by Nivre et al. (2007), and used it to produce preliminary trees. For each document, one annotator was given the preliminary trees to inspect and correct, with all dependencies visually marked so that parts of the sentence already inspected could easily be distinguished from those still awaiting inspection. The other annotator annotated the same document from scratch.

We have previously evaluated the effect of this protocol on annotation accuracy and speed (Haverinen et al. 2010b). We found that while a beginning annotator could benefit from a starting point, and a very experienced annotator could gain on speed while suffering a minor penalty on accuracy, on the whole, the preannotation protocol had no notable effect on the annotation. Based on this finding, the topic of preannotation will not be further pursued in this paper.

4.3 The second annotation layer

The second annotation layer was annotated by pairs of annotators out of a total of six annotators. In this layer, we used the existing tree structures annotated in the first phase as a starting point, and used a preprocessing software to suggest which additional dependencies could be present in the current sentence. The annotator was required to either confirm the dependencies or delete them, and in the case of relativizers, select their type. The deletion possibility was necessary, even for relativizers although they always have a secondary function, since the suggested dependency was not necessarily between the correct words. In the case of coordination propagation, the annotator could select a dependency governing or depending upon the head of a coordination structure and select whether this dependency propagates with respect to this coordination or not. In addition, it was possible to add or remove second layer dependencies manually, or change their types. This was sometimes necessary, as in the coordination propagation, dependencies occasionally modify some but not all coordinated elements or change their type while propagating. The first layer dependencies were not allowed to be modified at this stage.

5 Morphological analyses

This section describes the morphological layer of TDT. As manual annotation is expensive, and as there exists a suitable open source tool that is able to produce non-disambiguated morphological analyses of high quality, we have sought ways to obtain the morphological analyses for the treebank by automatically disambiguating the readings provided by this existing tool. The disambiguation is performed using machine learning, with unambiguous tokens serving as training data and syntactic analyses providing features. This Section begins by describing the tool used as the basis of the morphological layer, then continues to the methods used to disambiguate its output.

5.1 OMorFi

The morphological analyses in TDT are based on the output of OMorFi (Pirinen 2008; Lindén et al. 2009), which is a recent open source morphological analyzer of Finnish and part of the Open Source Morphologies (OMor) project by the University of Helsinki. As mentioned in Sect. 1, out of the open source tools for Finnish morphology, OMorFi has the best vocabulary coverage.

For each token, OMorFi returns a set of *morphological readings*. Each reading consists of a lemma (baseform) and a set of morphological tags, such as the main part-of-speech (POS), case, number, tense and so forth. A word can have multiple readings, in which case OMorFi generates all possible readings without disambiguation. Table 4 illustrates the OMorFi output.

OMorFi is able to produce different combinations of a total of 109 tags. In the figures given in this Section, we disregard numerals and punctuation, only considering word-like tokens. Out of all such tokens, 5.2 % are unknown to OMorFi, and approximately 46.1 % are unambiguous while the remaining 48.6 % receive more than one reading from OMorFi. On average, OMorFi assigns a token 1.8 readings, and for ambiguous tokens only, the average is 2.7 readings.

In order to be able to use the OMorFi output as the morphological analyses of the treebank, two main issues need to be addressed. First, a separate disambiguation step is needed for the 48.6 % of tokens in TDT that are ambiguous. Second, the unrecognized tokens (5.2 %) must be either manually annotated or addressed by other means. The specific methods used for post-processing the OMorFi output,

Table 4 OMorFi output

Word	Lemma	Translated lemma	POS	Other tags
Hän	*hän*	*he/she*	*Pron*	*Pers Sg Nom Up*
ei	*ei*	*not*	*V*	*Neg Sg3 Act*
asu	*asua*	*to live*	*V*	*Prs Ind ConNeg*
	asua	to live	V	Sg2 Act Imprt
	asu	outfit	N	Sg Nom
pienessä	*pieni*	*small*	*A*	*Sg Ine Pos*
kylässä	*kylä*	*village*	*N*	*Sg Ine*
	kylässä	visiting	Adv	

The correct readings are marked by emphasis. The example sentence can be translated as *He doesn't live in a small village*

treating unknown tokens and disambiguating between readings are discussed in the next subsections.

5.2 Post-processing OMorFi output and treatment of unknown tokens

In order to facilitate automatic disambiguation of the readings given by OMorFi, we have taken steps to post-process the OMorFi output to be more suitable for this purpose. In many cases, OMorFi produces multiple readings that are, for practical purposes, especially parsing, equivalent. This happens frequently with derivations and compounds, as Table 5 illustrates. As can be seen from the Table, the noun *tekeminen* (*doing*) is given two readings, one indicating that *tekeminen* is a minen-derivation (close to the ing-participle in English) of the verb *tehdä* (*to do*), and the other analyzes it directly as a noun. Even given context, it is not possible to judge which one of these two readings would be correct, as it can quite plausibly be claimed that both of them are. The same is true for the noun *isoisä*, which receives

Table 5 Examples of practically equivalent readings by OMorFi

Word	Lemma	Translated lemma	POS	Other tags
tekeminen	tekeminen	doing	N	Sg Nom
	tehdä	do	N (V)	Der_minen Sg Nom
isoisä	iso|isä	grand|father	A + N	Pos Sg Nom Cmpnd + Sg Nom
	iso|isä	grand|father	N + N	Pfx Cmpnd + Sg Nom
	iso|isä	grand|father	N + N	Sg Nom Cmpnd + Sg Nom
	isoisä	grandfather	N	Sg Nom

Top: *tekeminen (doing)*. The two readings only differ in whether tekeminen is marked as a derivation or a simple noun. Bottom: *isoisä (grandfather)*. There are four readings. The top reading suggests that *isoisä* is a compound of the adjective *iso (big)* and the noun *isä (father)*. According to the second reading, *iso* is a prefix for *isä*, and according to the third reading, the word is a compound of two nouns. The fourth and final reading analyzes *isoisä* as a simple noun. Note that due to being a derivation, the word *tekeminen* receives two POS tags from OMorFi, meaning that it has been derived from a verb, and the final wordform, the derivation, is a noun. In compound readings, both parts of the compound receive their own POS, marked with "+"

four readings from OMorFi. It can be analyzed as a compound of an adjective and a noun, a noun with a noun prefix, a compound of two nouns, or a simple noun. For practical purposes, all of these readings could be claimed to be equally plausible.

From the point of view of disambiguating OMorFi output using machine learning, it is not desirable that tokens have multiple readings that can all be judged correct. Thus we have, as a separate post-processing step, used rules to prune out extraneous readings. With readings that could potentially be analyzed as derivations, we have selected the direct, non-derivational alternative. With compounds, we have selected the simple noun reading, or in cases where all readings were compounds, the reading with the longest non-compound lemma.

In addition to removing unnecessary readings, we have addressed the issue of tokens that OMorFi does not recognize. There were in total 8,812 unrecognized tokens in TDT, together with the additional 706 null tokens inserted during syntax annotation, which were naturally not recognized either. For unknown tokens, we have used the standard set of OMorFi tags added with new tags for symbols, foreign words, typographical errors and colloquial words.

The treatment of unknown tokens consists of three phases. In the first phase, we considered unknown tokens that were compounds constructed using a dash, such as *Alzheimer-projekti (Alzheimer-project)*. For these tokens, we separated the parts of the compound and re-analyzed the latter part (*projekti*, *project*) with OMorFi, as in Finnish the last part of a compound dictates its category. If the latter part alone received an analysis, this analysis was kept and the word was given a lemma consisting of the first part unchanged and the lemma of the latter part as analyzed by OMorFi. In the second phase, regular expressions were used to find tokens that were in fact not words but rather symbols.

Finally, nearly all remaining unknown tokens were annotated manually. In the manual annotation, readings were given to unknown tokens either directly by an annotator, or in some cases by giving the word a model wordform, which was used to automatically acquire readings from OMorFi. Tokens that were covered by a *name* dependency (see Sect. 3.1) but that did not have any other syntactic structure marked were not, unlike other remaining unknown tokens, annotated manually. These tokens were automatically given the morphological tags of the head of the named entity, which in turn was manually annotated. The reasoning behind this is that these cases are very likely to consist of either foreign words, abbreviations or symbols, and the analysis is likely to stay the same throughout the whole named entity. Naturally, a named entity may consist of words of different POS, but these named entities are considered to be single units, where the internal analysis is irrelevant, as entities consisting of foreign words or symbols are not analyzed in the syntax, either. For the same reason, only the main tags, such as the main POS, are inherited and more fine-grained tags, such as those indicating case and number, are not. This strategy bears some resemblance to the Penn Treebank POS guidelines (Santorini 1990), according to which if a string of words is capitalized as a name (as in for instance *New York*), all words capitalized should be tagged as proper nouns, regardless of their actual POS. Figure 12 illustrates analysis inheritance in named entities consisting of unknown words.

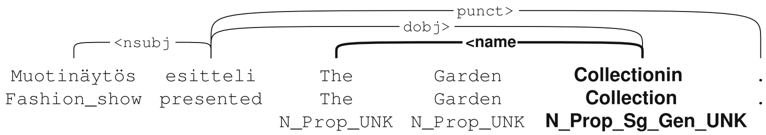


Fig. 12 Unknown words under a *name* dependency can inherit the morphological analysis of the governor, given that there is no internal structure annotated for the named entity. Note how only the main categories, N (noun) and Prop (proper) are inherited, and tags signaling less important features such as number and case are not. The tag UNK (unknown) denotes that the token was not recognized by OMorFi. The example sentence can be translated as *The fashion show presented The Garden Collection*

As mentioned above, also null tokens added during the syntax annotation are naturally not recognized by OMorFi, and thus they, too, require special attention with regard to their morphological analysis. The null tokens were manually annotated so that an annotator was given each null token with its context and instructed to assign it a model wordform, that is, a wordform that would most naturally fit in place of the null token. This wordform was then given to OMorFi to gain a reading or possibly several readings for the null token. Null tokens are not given a lemma, but otherwise they receive a full morphological analysis. In some cases a null token in fact represents several tokens, for instance a verb and its auxiliaries, in which case it receives the morphological analysis of the head word of the phrase it represents.

In addition to pruning out unnecessary readings and annotating unknown tokens, we have slightly modified the regular tagset assigned by OMorFi. As mentioned above, we have added new tags specific to unknown tokens. In addition to these tags, we have also added the tag *C* (*conjunction*), which is the POS for both subordinating and coordinating conjunctions. In some cases, OMorFi fails to assign the correct subcategory for adpositions and conjunctions. Thus, in order to avoid conflicting syntax and morphology annotations, we have added *Preposition* (*Pr*) and *Postposition* (*Po*) readings for every adposition, and *Subordinating conjunction* (*CS*) and *Coordinating conjunction* (*CC*) readings for every conjunction. We have also merged some tags into one to avoid assigning readings inaccurately in cases where evidence in the text is scarce. Such tags are *PxSg3* and *PxPl3*, which represent the third person singular and plural possessive suffixes. We have discarded the number information and used the tag *Px3* for both suffixes. Particles are also merged together with adverbs, and only the adverb tag *Adv* is used. Finally, the tags for different capitalizations, *cap*, *Cap* and *CAP* have been replaced with the single tag *up* signaling a capitalized word. Due to the pruning and modifications, there are tags that appear in the OMorFi output, but not in the final treebank. The total amount of different morphology tags in the final treebank is 107, as opposed to the 109 tags in the unprocessed OMorFi output.

After these post-processing steps, the numbers of readings have been reduced. Due to the manual annotation, there are no longer unrecognized tokens. In the automatically post-processed OMorfi output, the proportion of unambiguous tokens is approximately 62.9 %, and on average, one word has 1.6 readings, while the average as calculated for ambiguous tokens only is 2.6 readings. After the post-

Table 6 The origins of the morphological analyses of TDT

(%)	OMorFi output	Post-processed
Unambiguous	46.1	62.9
Ambiguous	48.6	37.0
Unknown	5.2	0.0

First, all tokens are given their analyses using OMorFi. Next, post-processing is used to reduce the amount of ambiguous tokens by pruning out unnecessary readings. In this phase, also unknown tokens are given analyses. Finally, all remaining ambiguity is resolved using a novel machine learning method

processing, 37.0 % of all tokens are still ambiguous, and these tokens require disambiguation using machine learning. Table 6 yet summarizes how the tokens of TDT receive their morphological analyses. The next subsection describes the machine learning method used for disambiguating between possible morphological readings.

5.3 Disambiguating OMorFi output as a machine learning task

In order to automatically disambiguate OMorFi output, we rely on two insights. First, barring exceptions discussed later in this section, morphological ambiguity is not systematic and depends on the specific wordform. Thus, for instance while the wordform *koirasta* is ambiguous between the singular partitive *koiras+ta* (*male*) and the singular elative *koira+sta* (*from dog*), most other nouns do not exhibit this ambiguity. In fact, as discussed in Sect. 5.2, 62.9 % of tokens in the treebank are unambiguous, and, in addition, even wordforms such as *koirasta* are often ambiguous only partially—here only the case and lemma are ambiguous. The second insight is that the existing syntactic annotation provides cues for morphological disambiguation. These cues can be direct, where a dependency type such as *aux* uniquely specifies the POS of the dependent, as well as indirect, where for instance the *nsubj* dependency type is mutually exclusive with a verbal reading of the dependent. Combining these two insights allows us to cast the disambiguation as a machine learning problem, where the partly or wholly unambiguous tokens serve as training examples, and the syntactic trees provide features for the disambiguation.

Modelling the task directly as a multi-class classification problem where each reading corresponds to one class is impractical, as there are 1,266 unique morphological tag combinations in the corpus. Rather, we cast the problem as a *ranking* of the alternative readings for every ambiguous token, where the highest-ranking reading is selected. In this approach, the reading is thus not a label to be predicted by a classifier, rather, it is used to generate features for the ranking. Throughout this section, a token will be considered unambiguous if it only has one reading, partially ambiguous if it has several possible readings all of which have the same main POS, and ambiguous otherwise.

All readings are represented using three general sets of features:

Reading features: For every unambiguous morphological category, i.e. a category whose value is the same for all readings of the token at hand, a binary feature is included specifying the category and its value. This includes also categories that are unambiguously absent, such as tense in nouns.

Syntactic features: The syntactic features are extracted from the complete dependency annotation, including the second layer where the analyses are not necessarily trees, and therefore a token can have more than one governor. For every governor, a binary feature is included that encodes the type of the dependency both alone and in combination with the type of the dependency for any of the governor's governors. Separate features are used when the token, or the token's governor, is the root.

Agreement features: Three morphological categories are directly relevant to agreement: person, number, and the possessive marker. For each of these categories, we define features that encode the agreement in the given category between the reading being ranked, and the readings of the token's governors and dependents. For every governor or dependent, and each of the three categories, a binary feature is emitted, encoding the dependency type (distinguishing between governors and dependents), the category in question, and the type of agreement. The agreement type compares the value of the category in the reading under consideration with the corresponding values in all readings of the governor or dependent. It can be *positive* if the value is equal in all readings, *possibly conflicting* if at least one reading explicitly disagrees with respect to the value (e.g. plural versus singular number), and *non-conflicting* if the values are not equal but do not explicitly disagree (e.g. plural versus unspecified number).

For computational reasons, we restrict the ranker to linear models. Under a linear model, the syntactic features as such do not contribute to the ranking as they are the same for all readings of any given token. We thus need to explicitly combine readings and syntactic features into feature pairs, a technique similar to polynomial kernel linearization in kernel-based classifiers. For every syntactic feature S_i and reading feature R_j, a new combined feature S_iR_j is thus introduced.

The final set of features representing each reading for ranking is the union of agreement features with the syntactic–reading feature pairs. The agreement features receive a constant weight of 1, while the weight of the combined syntactic–reading features is calculated as the pointwise mutual information of the two constituent features,

$$w(S_iR_j) = log\frac{P(S_i, R_j)}{P(S_i)P(R_j)}, \tag{1}$$

which has proved in our preliminary experiments to increase ranking accuracy over a simple constant weight.

The ranking is performed using a ranking support vector machine (SVM), implemented in the SVM^{rank} package (Joachims 2006). The ranking SVM learns a linear combination of features much like the commonly used linear SVM classifier would, but allows a *query structure* to be specified at training time. Only instances belonging to the same query are compared among each other. In our case, instances generated from the readings of a single token form one query. The training data consist of example queries with their correct ranking, here $+1$ for the correct reading and -1 for all other readings.

There are no manually annotated training data in the present setting and only unambiguous, or partially ambiguous tokens can be used for training. These, in turn, have no "negative" lower rank instances, which is exactly why they are unambiguous to begin with. However, recalling that the same reading with a different lemma may be ambiguous, depending on the wordform, we generate artificial negative examples from the three readings most often conflicting with the current positive instance (regardless of the lemma). For example, a singular partitive reading can be added as a negative example to the unambiguous singular elative *ikkunasta (from window)*, since this ambiguity is observed in wordforms such as *koirasta (from dog)* discussed above. The unambiguous singular elative reading is given the rank $+1$ and all artificial negative examples are given the rank -1, together forming one training query. This procedure is easily extended to partially ambiguous tokens, where each reading is given the rank $+1$ and contributes three artificial negatives with the rank -1 to the training query. Both unambiguous and partially ambiguous tokens thus serve as training data.

There is a small number of problem cases stemming from consistently ambiguous forms which do not lend themselves to the machine learning approach described so far. In English, an example of such a consistent ambiguity would be the verb infinitive, imperative, and present indicative (except for the 3rd person), all three of which systematically have the same wordform regardless of the lemma. This, in turn, means that there are no examples of this ambiguity for the ranker to learn from. The surrounding syntactic annotation, however, still provides cues for the disambiguation. We therefore develop a set of 11 rules to address the most common consistently ambiguous forms. The resulting disambiguation procedure is thus a hybrid approach where an initial machine learning output is post-processed with a small set of rules for specific, hard-to-learn cases.

The development of the machine learning method as well as the post-processing rules was carried out on a development set of 413 tokens with manually annotated morphological analyses. This set was also used for optimizing the regularization parameter C of the rank SVM. Since the ranker does not use any manually annotated morphological data, unambiguous and partially ambiguous tokens from the entire treebank were used in its training. Evaluation was performed on a set of 1000 manually annotated tokens from the test section of the treebank, as will be discussed in Sect. 6.2.

6 Evaluation

In this section, we evaluate the treebank annotation. We begin by evaluating the accuracy of the syntax annotation, and in the second subsection, we discuss the quality of the morphological analyses.

6.1 Syntax annotation quality

Our evaluation of the syntax annotation is twofold. As the first and second layers of the treebank were annotated in separate steps, the evaluation of these two tasks is performed in two steps as well.

In order to evaluate the quality of the first layer of annotation, we measure the performance of the annotators, henceforth called *annotator accuracy* (AA), using *labeled attachment score* (LAS), which is defined as the proportion of tokens, out of all tokens, that have been assigned the correct governor and dependency type. The measurement is made between an *individual* annotation and the *merged* annotation, not the *final* annotation. This is for two reasons, which stem from the annotation process described in Sect. 4. First, we want to avoid penalizing an annotator for a decision that was correct at annotation time but that has since then become outdated due to slight changes in the annotation scheme. Second, due to the tokenization and sentence splitting corrections made between the *merged* and *final* annotations, the number of tokens and sentences may differ between the *individual* and *final* annotations, which means that these annotations are no longer directly comparable.

The overall AA across the treebank sections and all annotators is 91.3 %. This gives us an estimate of the overall quality of the *individual* annotations. Table 7 lists annotator accuracy figures per annotator and per section. From these figures it can be seen that the quality of the single-annotation is high overall, the annotators are sufficiently trained and the annotation scheme is stable. Differences between treebank sections were rather small, in fact the differences between annotators and possibly a learning effect seem to influence the overall results more.

The overall AA and the figures in Table 7 describe the quality of the *individual* annotations, but not directly the quality of the double-annotated treebank. Therefore, we have conducted a small-scale experiment in order to evaluate the

Table 7 AA per annotator and per section

	Ann. 1	Ann. 2	Ann. 3	Ann. 4	Ann. 5	Overall
Wikipedia	95.7	85.1	90.4	–	94.5	91.1
Wikinews	95.5	87.8	92.4	74.3	–	91.1
Uni. news	96.6	89.5	92.0	70.6	–	88.6
Blogs	95.1	86.9	–	–	89.4	90.6
Student	95.4	86.2	88.6	72.4	–	88.6
Grammar	96.0	88.6	89.2	–	89.1	91.4
Europarl	96.0	88.1	92.5	74.6	88.9	91.8
JRC-Acquis	95.7	89.7	89.1	–	88.7	91.3
Financial	97.3	91.7	–	–	94.1	94.4
Fiction	96.2	88.9	–	–	91.8	92.6
Overall	95.9	88.0	90.5	71.8	90.6	91.3
Total annotated (%)	38.3	32.5	9.9	2.6	16.7	100.0

The row entitled *total annotated* gives the percentages of tokens annotated by each annotator, the total being twice the size of the corpus due to each token being annotated twice quality of the *final* annotation; a similar experiment was previously presented in a conference paper by Haverinen et al. (2011). The basic idea of this experiment was to evaluate a sample of the treebank by having an expert annotator annotate it for a third time, settle the divergences between the *final* annotation and the new annotation, and measure the LAS of the *final* annotation. When sampling the sentences to be annotated, sentences previously annotated by the same annotator should be avoided. This experiment setting assumes that the more proficient an annotator is, the more mistakes he or she will uncover from the original annotation.

However, it was not sufficient to measure the quality simply by letting Annotator 1 annotate a sample, because this annotator has previously annotated a very large portion of the treebank. Hence it was necessary to let Annotator 5, the second-best according to AA, evaluate the portion of the treebank previously annotated by Annotator 1. This strategy, naturally, leaves unevaluated the portion of the treebank (38,085 tokens, approximately 18.6 %) that was annotated by both Annotator 1 and Annotator 5. As these two annotators are the best-performing ones, the unevaluated section is likely the one with the highest accuracy, and thus this method of evaluation produces a conservative estimate of the quality. On the other hand, it is naturally possible that some errors go unnoticed due to three different annotators producing the same, erroneous analysis.

According to the strategy described above, Annotator 1 and Annotator 5 received a set of 200 randomly selected sentences each. These two annotators independently annotated their respective sentences, and a regular meeting of all annotators was then arranged to resolve the differences between the new annotation and the *final* annotation of the treebank. Effectively, by this we gained a triple-annotated set of 400 sentences, out of which 200 represent the set of sentences not annotated by Annotator 1, and the remaining 200 sentences represent the set of sentences annotated by Annotator 1 but not annotated by Annotator 5.

We have measured the annotator accuracy of the *final* annotation against the newly merged triple-annotated sample, weighted by the sizes of the portions which the two samples represent in the treebank. We find that the weighted LAS of the *final* annotation is 97.6 %. This figure gives an estimate of the quality of the *final* annotation, and together with the original overall annotator accuracy of 91.3 % it shows that full double-annotation is a thorough way to weed out errors; approximately 72 % [(97.6 − 91.3)/(100 − 91.3)] of the errors remaining in the single-annotated documents are eliminated by using the double-annotation protocol.

The second layer of the annotation has been evaluated in a slightly different manner. Due to the nature of the task, that is, annotating only a limited range of phenomena, a large number of the treebank tokens are completely irrelevant to the second layer annotation. Therefore, if we were to use LAS to measure annotator performance, this would result in artificially high figures due to an overwhelming amount of tokens being trivially correct. Therefore, for evaluating the second layer of annotation, we use the F_1-score, defined as $F_1 = \frac{2PR}{P+R}$, where P (precision) is the fraction of dependencies in the evaluated output that are present in the gold standard, and R (recall) is the fraction of dependencies in the gold standard that are present in the evaluated output. Table 8 gives the F_1-scores for each of the six annotators participating in the second layer annotation.

Table 8 Evaluation of the second annotation layer given in precision (P), recall (R) and F_1-score

Annotator	P	R	F_1
Ann. 1	98.2	97.5	97.8
Ann. 2	96.6	96.0	96.3
Ann. 3	95.3	95.5	95.4
Ann. 5	98.2	97.7	97.9
Ann. 6	95.0	93.4	94.2
Ann. 7	94.9	92.1	93.5
Overall	96.7	95.8	96.3

As can be seen from these figures, the annotation quality for the second annotation layer is consistently high. In fact, although the two measures are not directly comparable, it would seem that the second layer annotation was the more straightforward of the two tasks, which is an intuitive result, given that in the second layer annotation, an annotator was not required to create a full tree structure, but rather decide whether a suggested dependency is present, and in some cases, what its type is.

6.2 Quality of the morphological analyses

In order to evaluate the accuracy of the disambiguation procedure described in Sect. 5.3, we have double-annotated the morphological analyses for 1,000 tokens from the treebank test section, sampling from all tokens except for punctuation and numerals. The morphology test set thus also includes null tokens and tokens unknown to OMorFi. On this set, we report the accuracy of the morphological analyses on three different granularities: the main POS, fine-grained tagging including POS and all other morphological tags, and full morphology, which includes fine-grained tagging

Table 9 Morphology assignment evaluation in terms of accuracy

	POS	POS+tags	POS+tags+Lemma
OMorFi 1 reading	97.6	96.5	96.3
OMorFi 2+ readings	95.6	91.1	90.1
OMorFi unknown/null token	100.0	94.4	94.4
All	96.7	93.7	93.1
All with punct./number	97.3	94.8	94.3

Results are given separately for tokens with only a single OMorFi reading, which do not undergo any disambiguation, tokens with more than one OMorFi reading, which are disambiguated using the procedure described in Sect. 5.3, and finally, tokens not recognized by OMorFi and null tokens, whose analyses are given manually. The *All* row shows the overall result on the test set, while the last row shows the overall accuracy for all tokens, including punctuation and numbers. Note that even without any disambiguation (the first row), the accuracy is not 100 %, due to cases where OMorFi analyzes a word erroneously as well as the word lemma. The results are shown in Table 9. The table lists separately results for unknown tokens and null tokens, which, as described in Sect. 5.2, also receive morphological analyses. The performance on all tokens including numbers and punctuation is estimated by an average accuracy weighted by the proportion of numbers and punctuation in the corpus, assuming that these tokens always receive the correct analysis.

Not all errors in the morphology assignment are due to the machine learning disambiguation method. There are also cases where OMorFi generates one or more readings for a token, but none of the readings are correct in the given context. Thus, we have separately evaluated the machine learning component of the morphology assignment, by ignoring the misassignments of OMorFi or in other words, cases where the gold standard analysis indicates that the correct reading is not one of those given by OMorFi. In this manner of evaluation, the accuracy of the main POS is 98.7 %, the accuracy of the fine-grained tagging 96.2 %, and the accuracy of the full morphology 95.7 %, calculated on the test set ignoring punctuation and numbers. Tokens originally unknown to OMorFi as well as null tokens are included in these figures, as they are given the correct analyses manually, and thus the disambiguation method has the valid options at its disposal. If we further restrict the data which the evaluation is performed on by disregarding all unambiguous tokens, as these tokens do not need the machine learning to receive a single reading, the main POS receives an accuracy of 97.5 %, the fine-grained tagging an accuracy of 93.0 % and the full morphology an accuracy of 92.1 %.

In addition to POS, the morphological tags assigned by OMorFi belong to 16 different categories. Table 10 presents results category by category in Precision, Recall and F_1-score. Two categories, *Casechange* and *Other*,[6] are not included in the table due to fact that these tags are manually added and therefore considered to always be correct. As can be seen from the figures, most, but not all, categories are predicted very accurately. The category with the lowest F_1-score is *Clitic*, largely due to the ambiguity between a plain adverb and an adverb with a clitic. A typical case is the wordform *ainakin*, which can be a plain adverb with the meaning *at least* or the adverb *aina* (*always*) with the clitic *-kin* (*also*). Both of these readings are adverb modifiers in the syntactic tree, meaning that the tree does not provide cues for their disambiguation.

7 Treebank data and associated tools

This section describes the totality of the data, related tools and aids that are released as parts of the contribution of this paper.

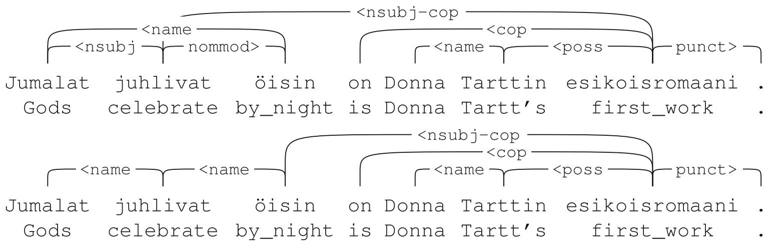
The Turku Dependency Treebank is released in its native xml-format. As described previously in Sect. 3.1, even the first layer of the syntax annotation in TDT does not in fact contain strict tree structures but rather directed graphs, due to the treatment of multi-word named entities. Therefore, we have transformed the

[6] The category Other contains tags that indicate tokens unknown to OMorFi, typographical errors and colloquial wordforms. OMorFi does not produce these tags.

Table 10 Precision (P), Recall (R) and F_1-score (F_1) given separately for each feature category in the morphology assignment

	P	R	F_1
Subcategory	95.2	95.2	95.2
Number	96.9	98.4	97.7
Case	97.1	98.4	97.7
Possessive suffix	94.1	100.0	97.0
Person	97.9	99.3	98.6
Voice	97.3	99.0	98.1
Tense	98.5	98.5	98.5
Mood	99.3	99.3	99.3
Negation	100.0	88.9	94.1
Participle	96.8	100.0	98.4
Infinitive	100.0	100.0	100.0
Clitic	76.9	90.9	83.3
Derivation	92.9	92.9	92.9
Comparison	93.9	95.7	94.8

The feature category named *Subcategory* contains tags that amplify the main POS, such as subordinating conjunction or coordinating conjunction for the main POS conjunction, or proper noun for the main POS noun



<nsubj-cop
<name
<nsubj
nommod>
<cop
<name
<poss
punct>
Jumalat juhlivat öisin on Donna Tarttin esikoisromaani .
Gods celebrate by_night is Donna Tartt's first_work .
<nsubj-cop
<cop
<name
<name
<name
<poss
punct>
Jumalat juhlivat öisin on Donna Tarttin esikoisromaani .
Gods celebrate by_night is Donna Tartt's first_work .

Fig. 13 The original *name* dependencies in TDT (*top*), and the *name* dependencies as processed in order to ensure treeness (*bottom*)

treebank so that it only contains trees and distribute this version in the commonly used CoNLL-09 format (Hajič et al. 2009). The transformation is performed by only including the first layer of annotation and additionally modifying the *name* dependencies. *Name* dependencies that cover multiple tokens are expanded to chains of *name* dependencies from right to left, and the rightmost token of the named entity becomes its head, meaning that the token governing the named entity is the governor of the rightmost token. If the named entity has an internal structure marked, this structure is deleted. The transformation of the *name* dependencies is illustrated in Fig. 13.

There are two ways of accessing the treebank: it can either be downloaded, or it can be browsed and queried directly online. Using the browseable version, it is possible to view sentences of the treebank in their document context. The search functions enable searching for wordforms, morphological features and syntactic structures.

In addition to the *final* annotations of the treebank, we make available the *individual* and *merged* annotations described in Sect. 4 for the first annotation layer.

This data is to our knowledge unique in that it allows one to study the annotation process, which is usually not possible, due to many treebanks being single-annotated after an initial annotator training phase.

As a technical aide, we provide a web-based parser evaluation service that measures parser output against the strict tree version of the test set of the treebank. The test set consists of 21,281 tokens (1,554 sentences), as selected in a random, stratified manner on the level of documents. The syntax of the test set in its entirety is manually double-annotated, and the morphology is manually double-annotated for a subset of 1000 tokens, the same subset that was used for evaluating the treebank morphology in Sect. 6.2. The service provides the user with a text-only version of the test set, in the CoNLL-09 format. This version, naturally, lacks the dependency analyses of the sentences, the null tokens, and the morphological analyses. A parsed version in the CoNLL-09 format can be submitted to the system, which will return evaluations of both the morphological tagging and the syntactic parsing. For the morphology evaluation, the system returns the accuracy of lemmatization, main POS, fine-grained tagging, as well as the full morphology. The syntax evaluation results, in turn, are given in labeled as well as unlabeled attachment scores, and dependency type accuracy. The frequency of use for the evaluation system is restricted to 10 submissions daily and 15 submissions weekly, in order to prevent overfitting of the test set.

The service can evaluate parser output with null tokens inserted, and in fact, in order to achieve perfect performance scores, a parser is expected to provide these tokens. The position of a null token in the sentence is not evaluated, as there are often multiple possible placements due to the free word order of Finnish. The service thus aligns all null tokens to their gold standard equivalents so as to maximize the LAS. For each missing null token in the parser output, all the dependents of the token are counted as having the incorrect governor, and any extraneous null token in the parser output has its governor and dependency type counted as incorrect. By submitting the test set otherwise fully correctly parsed, but with null tokens omitted, it is possible to reach a labeled attachment score of 99.04 %, the unlabeled attachment score being exactly the same. In terms of morphology evaluation, a submission without null tokens can receive an accuracy of 99.5 % in lemmatization, POS tagging, fine grained-tagging and full morphology alike.

The syntactic analysis is evaluated on the full test set, whereas the evaluation of the morphological analysis is carried out on the 1000-token subset that has morphological gold standard annotation available. For a token to be considered correct for a metric, all values relating to this metric must be correct. Accuracy is calculated as the percentage of correct tokens out of all tokens evaluated. Some of the null tokens of the test set are also part of the morphological gold standard, and the aligned null tokens in the parser output will be evaluated against these null tokens in the morphological evaluation. If no aligned null token is found, the morphological evaluation considers the token incorrect.

This section concludes the discussion of the treebank. Next, we move on to describe the second main contribution of the paper, the freely available statistical parsing pipeline of Finnish.

8 Statistical dependency parsing of Finnish

The initial main motivation for the development of the Turku Dependency Treebank was the need for training data for statistical dependency parsing of Finnish. In this section, we introduce a full parsing pipeline trained on the treebank, using state-of-the-art open source tools. This pipeline constitutes the first freely available dependency parser for Finnish and can be used as a starting point for further research in Finnish dependency parsing, as well as incorporated in various NLP tasks and applications. The first such applications are briefly introduced in the second subsection.

8.1 Parsing pipeline

The parsing pipeline follows the "standard" task sequence of sentence splitting, tokenization, morphological tagging, and dependency parsing. Sentence splitting and tokenization are machine-learned using the corresponding modules from the Apache OpenNLP toolkit.[7] Dependency parsing of the morphologically tagged input is carried out using the graph-based parser of Bohnet (2010), a state-of-the-art statistical dependency parser. These components are used off-the-shelf and trained in a standard manner without any adaptation.

Morphological tagging, on the other hand, requires more effort before sufficient accuracy can be gained. In a number of preliminary experiments, we were unable to achieve an acceptable parsing performance using purely machine-learned taggers and lemmatizers, with a strong indication that the poor performance of statistical lemmatization in particular was responsible for the low overall parsing accuracy. We thus implement morphological tagging as a hybrid system combining the OMorFi analyzer with the HunPOS statistical tagger (Halácsy et al. 2007), an open source reimplementation of the TnT tagger of Brants (2000). The combination of these two tools is rather straightforward, since the HunPOS tagger allows one to list the possible readings for any token in the input. This set of readings is then used to constrain the search space during disambiguation. For each token that is recognized by OMorFi, the set of possible readings is passed to HunPOS, which will select one of them. Tokens which are not recognized by OMorFi are left for HunPOS to tag using suffix-guessing. An important aspect of this hybrid approach is that for tokens recognized by OMorFi, which are the majority of running tokens, correct lemmas are obtained as well. For the tokens not recognized by OMorFi, we find that the best strategy is not to attempt any lemmatization and to set the lemma to be the token itself.

The employed tools have several optimizable parameters: in HunPOS the transition and emission probability order as well as two parameters governing suffix guessing, and in the dependency parser the weight vector size and non-projectivity threshold. The parameters were optimized separately for each tool using a grid search evaluated on a held-out development set. A joint search across the pipeline is computationally infeasible due to the long training times of the dependency parser.

[7] http://opennlp.apache.org/

The optimal parameter combination was then used to train the final models, using a union of the training and development data. The test data was not used at any point during the parameter optimization, nor was it used during the overall development of the parsing pipeline.

The overall performance of the parser is evaluated using the test set evaluation service described in Sect. 7 and is summarized in Table 11. The labeled and unlabeled (UAS) attachment scores are 81 and 85 % respectively, including the approximately −1pp penalty incured on the test set by any parser that is not capable of introducing the null tokens. Per-section results are shown in Table 12. Here we see a wide variance in parsing performance, from 74.5 % LAS on fiction to 88 % LAS on the JRC-Acquis legal documents.

In order to set these results in a wider context, we have collected parsing results for a variety of languages from a range of recent studies, including the CoNLL'09 shared task (Hajič et al. 2009), the study presenting the MateTools parser used in this work (Bohnet 2010), and the studies by Nivre et al. (2007), Nivre (2008) and Farkas et al. (2012). In Fig. 14, we compare the Finnish parsing performance to the results presented in these studies. Taking into account the size of the corpus, the parsing accuracy is in the expected range.

Table 11 Dependency parser results on the test set

Metric	Values tested	Accuracy (%)
Labeled attachment score (LAS)	Governor + Dependency type	81.01
Unlabeled attachment score (UAS)	Governor	84.97
Dependency type accuracy	Dependency type	89.53
Lemmatization	Lemma	91.8
Main part-of-speech tagging	POS	94.4
Fine-grained tagging	All morphological tags	89.8
Full morphology	Lemma + all morphological tags	87.3

Table 12 Per-section dependency parser results on the test set, given in labeled (LAS) and unlabeled (UAS) attachment scores

Section	LAS	UAS
Wikipedia	82.71	86.97
Wikinews	80.33	85.23
Uni. news	84.65	88.50
Blogs	76.84	80.45
Student	82.65	87.18
Grammar	79.59	83.83
Europarl	83.14	86.52
JRC-Acquis	88.01	90.98
Financial	79.93	82.48
Fiction	74.47	79.35

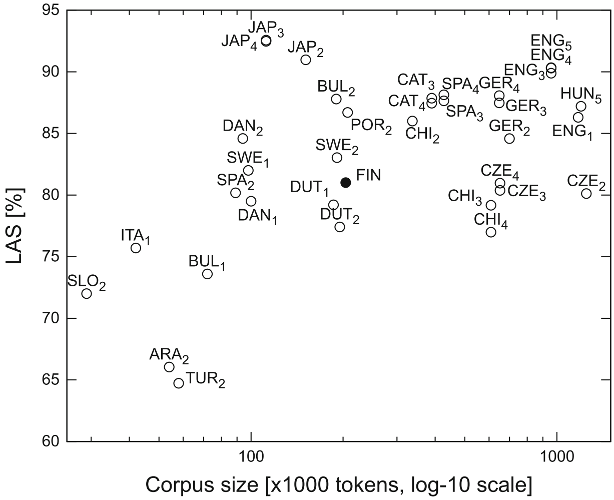


Fig. 14 Parsing results for various languages from a number of studies, as related to corpus size. The languages are given as short labels, where *CAT* Catalan, *CHI* Chinese, *CZE* Czech, *ENG* English, *GER* German, *JAP* Japanese, *SPA* Spanish, *ARA* Arabic, *BUL* Bulgarian, *DAN* Danish, *DUT* Dutch, *POR* Portuguese, *SLO* Slovene, *SWE* Swedish, *TUR* Turkish, *HUN* Hungarian, *ITA* Italian and *FIN* Finnish. The different studies are indicated as subscripted numbers as follows: *1* Nivre et al. (2007), *2* Nivre (2008), *3* Hajič et al (2009), *4* Bohnet (2010) and *5* Farkas et al. (2012). The result achieved by the parser presented in this work is shown as a *black dot*

8.2 Existing applications

Even while under development, earlier versions of TDT and the parsing pipeline have so far been applied in several projects, demonstrating their utility in Finnish NLP.

As part of the FinCLARIN consortium, the treebank and the parser were used to produce a parsebank of the Finnish sections of the Europarl (Koehn 2005) and JRC Acquis (Steinberger et al. 2006) corpora, in the FinnTreeBank syntactic scheme. The result of this project is the parsebank distributed as FinnTreeBank 3, as described by Voutilainen et al. (2012a).

TDT has also been used in a project conducted in collaboration with the machine translation company Convertus AB.[8] Part of the *Bologna project*, this project builds a machine translation system from Finnish to English focusing on the educational domain.

9 Comparison of TDT and FinnTreeBank

As mentioned in Sect. 1, FinnTreeBank (Voutilainen and Lindén 2011) is a second publicly available Finnish treebank, published shortly after the second intermediate

[8] http://www.convertus.se/.

release of the Turku Dependency Treebank. This section discusses the differences between the two treebanks, which are summarized in Table 13.

The first difference is that FinnTreeBank has been created as a grammar definition corpus, allowing the construction of a rule-based parser, whereas TDT has been built with statistical parsing in mind. FinnTreeBank contains text from four genres: example sentences from the Finnish grammar (Hakulinen et al. 2004), Wikipedia, online news (from *Helsingin Sanomat* and *Tietoviikko*) and fiction (a sample of the Finnish translation of Jostein Gaarder's *Sophie's world*). The latter three genres constitute approximately 3 % of FinnTreeBank, while the grand majority of the treebank, almost 97 %, are grammar examples. As described in Sect. 2, TDT has ten different genres. We have selected grammar examples from FinnTreeBank as one section of TDT, so as to enable a conversion between the schemes of the two treebanks, as discussed below. The size of FinnTreeBank is 169,450 tokens (19,764 sentences), that is, approximately 30,000 tokens smaller than the Turku Dependency Treebank. The sentence count of FinnTreeBank is larger than that of TDT, which is due to the most common genre of the treebank being grammar examples, which are often rather short.

In addition to size and genres, the two treebanks differ in several other respects. First, the annotation schemes are different. While TDT uses a modified version of a previously existing scheme, the Stanford Dependency scheme (49 dependency types), FinnTreeBank is annotated in a custom annotation scheme, henceforth called the FTB scheme. This scheme contains 14 dependency types, disregarding the type *main* which marks the main predicate of the sentence. Table 14 lists the dependency types of the FTB scheme. The SD scheme is more detailed with its treatment of several phenomena, making distinctions which the FTB scheme does not explicitly make. For instance, the SD scheme distinguishes between different kinds of noun premodifiers, such as adjectival modifiers, genitive modifiers and determiners, whereas the FTB scheme analyzes all of these as attributes. Similarly, the FTB scheme analyzes nominal modifiers after a noun as well as full relative clauses as simply postmodifiers, whereas in the SD scheme these are distinguished using different dependency types.

Second, FinnTreeBank is, for the most part, single-annotated (a small portion of 2039 tokens has been double-annotated (Voutilainen and Purtonen 2011)), while, as discussed in Sect. 4, the annotation protocol in TDT is full double-annotation.

Table 13 Comparison of the main features of the Turku Dependency Treebank and FinnTreeBank

	Turku Dependency Treebank	FinnTreeBank
Size in tokens	204,399	169,450
Size in sentences	15,126	19,764
Genres	10	4 (97 % grammar examples)
Annotation scheme	Stanford Dependency	FTB scheme
Dependency types in scheme	49	14
Annotation protocol	double	(mostly) single
Morphology annotation	OMorFi disambiguated	3 taggers

Table 14 Dependency types of the FTB scheme

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

Finally, the treebanks differ in their morphological analyses. The analyses in FinnTreeBank, like those in TDT, are based on the tagset of the automatic tool OMorFi. According to the manual of Voutilainen et al. (2012b, p. 8), the morphological analyses have been created by manually checking the combined output of three different statistical taggers. In TDT, as described in Sect. 5, the morphological readings have been disambiguated semi-automatically based on the manual syntax annotation.

As part of the FinCLARIN parsebank project mentioned in Sect. 8.2, we have converted an earlier version of TDT, consisting of 190,271 tokens (93 % of the final size), into the FTB scheme. Unlike in the current treebank, the morphology information in this project was provided through the commercial FinCG (Karlsson 1990) analyzer by Lingsoft Inc. We used the converted version of the treebank to train an earlier version of the statistical parser of Bohnet (2010). This parser was used to parse a corpus of Finnish consisting of 76.3M tokens from the Europarl (Koehn 2005) and JRC-Acquis (Steinberger et al. 2006) corpora. This parsebank is distributed by the University of Helsinki as FinnTreeBank 3 (Voutilainen et al. 2012a). The conversion of SD into the FTB scheme was mostly rule-based, with a machine-learning post-processing component that connected islands left in the converted output after the rule-based step. The conversion relies on the FinCG output and its detailed description is beyond the scope of this paper.

As part of this project, we made a focused effort to pool TDT with FinnTreeBank so as to leverage the larger combined training set size. We were, however, unable to increase parsing performance, likely due to the combination of errors introduced during the necessary scheme and morphology transformations, as well as the fact that FinnTreeBank is not developed for statistical parser training and its domain of grammar examples is very specific.

10 Conclusion

In this paper, we have presented the Turku Dependency Treebank, a publicly available dependency-based treebank of Finnish. Prior to the earlier, smaller versions of this treebank, Finnish did not have such a resource available, hindering research in many areas of statistical NLP as well as preventing the development of freely available, open source statistical dependency parser. The treebank consists of 204,399 tokens (15,126 sentences), and it contains text from ten different sources. As the second main contribution of the paper, we have presented a freely available statistical parsing pipeline of Finnish. The treebank and the parsing pipeline alongside with other related tools and resources are publicly available under the *Creative Commons Attribution-Share Alike* license at http://bionlp.utu.fi/.

The Turku Dependency Treebank contains gold standard syntax annotations in the widely used Stanford Dependency (SD) scheme as well as automatically assigned and fully disambiguated morphological analyses. The syntax annotation of the treebank contains not only the base-syntactic layer that is grounded on the *basic* variant of the SD scheme, but also a second layer, termed *coordination propagation and additional dependencies*, which provides information about coordination structures, open clausal complements and relative clauses. To our knowledge, TDT is the first resource manually annotated for the coordination propagation contained in the SD scheme.

The morphological analyses of the treebank are based on the output of an existing tool, OMorFi. In order to produce high-quality morphological analyses in an efficient manner while avoiding costly manual annotation, we have used a novel method to disambiguate the readings assigned by OMorFi. This method employs machine learning, using the unambiguous tokens of the treebank as training examples and the syntactic annotation for providing features. While manually annotating morphology is, naturally, the most precise manner of producing these analyses, our method is more cost-efficient as the manual effort is minimized and the existing syntactic trees are used to aid the disambiguation.

The annotation protocol of the treebank in its entirety is full double annotation, which enables us to ensure high quality annotations, as demonstrated by a high overall annotator accuracy of 91.3 % in LAS. This evaluates the quality of the *individual* annotations, while the quality of the *final* annotation of the treebank is evaluated in a separate experiment, where a portion of the treebank was triple-annotated. The LAS of the *final* annotation against the triple-annotated gold standard was 97.6 %. The annotator accuracy of the second annotation layer, where the structures annotated are not full trees, was measured in F_1-score, and the resulting overall AA was 96.3 %. The evaluation of the morphological analyses showed that the automatic method for morphology assignment achieves an accuracy of 96.7 % in main POS and 93.1 % in full morphological analyses, including fine-grained tagging and lemmatization. This goes to demonstrate that good quality morphology disambiguation can be achieved automatically, without any manually annotated training data specific to the task.

In addition to the *final* annotations of the treebank, we also release the *individual* and *merged* annotations, which are, to our knowledge, a unique resource for studying the human annotation process. We also provide a web-based service for evaluating parsers on our treebank test set.

As the second part of the main contribution in this work, we provide a full statistical parsing pipeline for Finnish, including a sentence splitter, a tokenizer, a morphological tagger and a parser. For this pipeline, we have induced a statistical parser of Finnish, using the state-of-the-art parser of Bohnet (2010). The parsing pipeline achieves a performance of 81 % in LAS, and is freely available alongside with the treebank.

In the future, it would be highly useful to further enhance the annotations of TDT. At the time of writing of this paper, we are in the process of annotating verb argument structures using the well-known PropBank scheme as originally described by Palmer et al. (2005). To improve the parsing performance achieved in this work, it would be helpful to increase the size of the treebank using single-annotation, possibly employing active learning (Cohn et al. 1996) to identify the most beneficial examples to annotate. Additionally, identifying and correcting the most common errors in the morphological layer of the treebank would be beneficial. Other possible future work directions include annotating argument structures of nouns in the NomBank scheme (Meyers et al. 2004) and parallelizing the treebank with another language for machine translation purposes.

Acknowledgements We would like to thank the authors of the treebank text, who kindly allowed us to use their work, either by explicit permission or by releasing their text under an open license originally. We would also like to thank Lingsoft Inc. for making FinTWOL and FinCG available to us throughout the project. This work has been supported by the Academy of Finland, Turun yliopiston Yliopistosäätiö, Emil Aaltosen säätiö and TOP-säätiö.

References

Begum, R., Dhwai, A., & Misra, D. (2008). Dependency annotation scheme for Indian languages. In *Proceedings of IJNLP'08*, pp. 721–726.

Björne, J., Ginter, F., Pyysalo, S., Tsujii, J., & Salakoski, T. (2010). Complex event extraction at pubmed scale. *Bioinformatics, 26*(12), 382–390.

Boguslavsky, I., Chardin, I., Grigorieva, S., Grigoriev, N., Iomdin, L., Kreidlin, L., et al. (2002). Development of a dependency treebank for Russian and its possible applications in NLP. In *Proceedings of LREC'02*, pp. 852–856.

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

Brants, S., Dipper, S., Hansen, S., Lezius, W., Smith, G. (2002). The TIGER treebank. In *Proceedings of TLT1*, pp. 24–41.

Brants, T. (2000). TnT—a statistical part-of-speech tagger. In *Proceedings of ANLP'00*, pp. 224–231.

Cer, D., de Marneffe, M. C., Jurafsky, D., Manning, C. (2010). Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC'10*, pp. 1628–1632.

Charniak, E., & Johnson, M. (2005). Coarse-to-fine n-best parsing and MaxEnt discriminative re-ranking. In *Proceedings of ACL'05*, pp. 173–180.

Choi, J. D., & Palmer, M. (2011). Getting the most out of transition-based dependency parsing. In *Proceedings of ACL-HLT'11*, pp. 687–692.

Clegg, A. B., & Shepherd, A. (2007). Benchmarking natural-language parsers for biological applications using dependency graphs. *BMC Bioinformatics, 8*(1), 24.

Cohn, D., Ghahramani, Z., & Jordan, M. (1996). Active learning with statistical models. *Journal of Artificial Intelligence Research, 4*, 129–145.

de Marneffe, M. C., & Manning, C. (2008a). Stanford typed dependencies manual. Tech. rep., Stanford University, revised for Stanford Parser v. 2.0.4 in November 2012.

de Marneffe, M. C., & Manning, C. (2008b). Stanford typed dependencies representation. In *Proceedings of COLING'08, workshop on cross-framework and cross-domain parser evaluation*, pp. 1–8.

Dukes, K., & Buckwalter, T. (2010). A dependency treebank of the Quran using traditional Arabic grammar. In *Proceedings of INFOS'10*, pp. 1–7.

Farkas, R., Vincze, V., & Schmid, H. (2012). Dependency parsing of hungarian: baseline results and challenges. In *Proceedings of EACL '12*, pp. 55–65.

Foster, J., Cetinoglu, O., Wagner, J., Le Roux, J., Nivre, J., Hogan, D., et al. (2011). From news to comment: Resources and benchmarks for parsing the language of web 2.0. In *Proceedings of IJCNLP'11*, pp. 893–901.

Hajič, J. (1998). Building a syntactically annotated corpus: The Prague Dependency Treebank. In *Issues of valency and meaning. Studies in Honour of Jarmila Panevová, Karolinum* (pp. 106–132) Prague, Czech Republic: Charles University Press.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., et al. (2009). The CoNLL-2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09*.

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

Halácsy, P., Kornai, A., & Oravecz, C. (2007). HunPos—an open source trigram tagger. In *Proceedings of ACL'07, Companion Volume*, pp. 209–212.

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

Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., Salakoski, T. (2009). Dependency annotation of Wikipedia: First steps towards a Finnish treebank. In *Proceedings of TLT8*, pp. 95–105.

Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., Salakoski, T. (2010a). Dependency-based propbanking of clinical Finnish. In *Proceedings of LAW IV*, pp. 137–141.

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

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

Helasvuo, M. L., & Huumo, T. (2010). Mikä on subjekti?. *Virittäjä, 114*(1), 165–195.

Joachims, T. (2006). Training linear SVMs in linear time. In *Proceedings of the ACM conference on knowledge discovery and data mining*.

Karlsson, F. (1990). Constraint grammar as a framework for parsing running text. In *Proceedings of COLING'90*, pp. 168–173.

Klein, D., & Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of ACL'03*, pp. 423–430.

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

Koskenniemi, K. (1983). Two-level model for morphological analysis. In *Proceedings of IJCAI'83*, pp. 683–685.

Lee, J., & Kong, Y. H. (2012). A dependency treebank of classical Chinese poems. In *Proceedings of NAACL-HLT 2012*, pp 191–199.

Lindén, K., Silfverberg, M., & Pirinen, T. (2009). HFST tools for morphology—an efficient open-source package for construction of morphological analyzers. In *State of the Art in Computational Morphology, Communications in Computer and Information Science, vol. 41*, pp 28–47.

Marcus, M., Marcinkiwicz, M. A., & Santorini, B. (1993). Building a large annotated corpus of English: The Penn Treebank. *Computational Linguistics, 19*(2), 313–330.

McDonald, R., Lerman, K., & Pereira, F. (2006). Multilingual dependency analysis with a two-stage discriminative parser. In *Proceedings of CoNLL'06*, pp. 216–220.

Meena, A., & Prabhakar, T. V. (2007). Sentence level sentiment analysis in the presence of conjuncts using linguistic analysis. In *Proceedings of ECIR'07*, pp. 573–580.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., et al. (2004). The NomBank project: An interim report. In *Proceedings of the NAACL/HLT workshop on frontiers in corpus annotation.*

Miwa, M., Pyysalo, S., Hara, T., & Tsujii, J. (2010). A comparative study of syntactic parsers for event extraction. In *Proceedings of BioNLP'10*, pp. 37–45.

Miyao, Y., Sagae, K., Sætre, R., Matsuzaki, T., Tsujii, J. (2009). Evaluating contributions of natural language parsers to protein-protein interaction extraction. *Bioinformatics, 25*(3), 394–400.

Nivre, J. (2008). Algorithms for deterministic incremental dependency parsing. *Computational Linguistics, 34*(4), 513–553.

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

Nivre, J., Rimell, L., McDonald, R., Gómez-Rodríguez, C. (2010). Evaluation of dependency parsers on unbounded dependencies. In *Proceedings of COLING'10*, pp. 833–841.

Palmer, M., Gildea, D., & Kingsbury, P. (2005). The Proposition Bank: An annotated corpus of semantic roles. *Computational Linguistics, 31*(1), 71–106.

Pirinen, T. (2008). Suomen kielen äärellistilainen automaattinen morfologinen jäsennin avoimen lähdekoodin resurssein. Master's thesis, University of Helsinki.

Qian, L., & Zhou, G. (2012). Tree kernel-based protein-protein interaction extraction from biomedical literature. *Journal of Biomedical Informatics, 45*(3), 535–543.

Santorini, B. (1990). Part-of-speech tagging guidelines for the Penn Treebank project. Tech. rep., University of Pennsylvania, (3rd revision, 2nd printing).

Seraji, M., Megyesi, B., & Nivre, J. (2012). Bootstrapping a Persian dependency treebank. *Linguistic Issues in Language Technology, 7*(18).

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

Tratz, S., & Hovy, E. (2011). A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of EMNLP'11*, pp. 1257–1268.

Valkonen, K., Jäppinen, H., & Lehtola, A. (1987). Blackboard-based dependency parsing. In *Proceedings of IJCAI'87—volume 2*, pp. 700–702.

Vincze, V., Dóra, S., Almási, A., Móra, G., Alexin, Z., & Csirik, J. (2010). Hungarian dependency Treebank. In *Proceedings of LREC'10*, pp. 1855–1862.

Voutilainen, A., & Lindén, K. (2011). Specifying a linguistic representation with a grammar definition corpus. In *Proceedings of corpus linguistics 2011.*

Voutilainen, A., & Purtonen, T. (2011). A double-blind experiment on interannotator agreement: The case of dependency syntax and Finnish. In *Proceedings of NODALIDA'11*, pp. 319–322.

Voutilainen, A., Muhonen, K., Purtonen, T., & Lindén, K. (2012a). Specifying treebanks, outsourcing parsebanks: Finntreebank 3. In *Proceedings of LREC'12.*

Voutilainen, A., Purtonen, T., & Muhonen, K. (2012b). FinnTreeBank2 manual. Tech. rep., University of Helsinki, Department of Modern Languages.

Zhuang, L., Jing, F., & Zhu, X. Y. (2006). Movie review mining and summarization. In *Proceedings of CIKM'06*, pp. 43–50.