Katri Haverinen

Natural Language Processing Resources for Finnish

Corpus Development in the General and Clinical Domains

Turku Centre *for* Computer Science

TUCS Dissertations
No 179, August 2014

Natural Language Processing Resources for Finnish

Corpus Development in the General and Clinical Domains

Katri Haverinen

To be presented, with the permission of the Faculty of Mathematics and Natural Sciences of the University of Turku, for public criticism in Auditorium Beta on September 4, 2014, at 12 noon.

University of Turku
Department of Information Technology
Joukahaisenkatu 3-5
20520 Turku
Finland

2014

Supervisors

Dr. Filip Ginter
Department of Information Technology
University of Turku
Joukahaisenkatu 3-5
20520 Turku
Finland

Professor Tapio Salakoski
Department of Information Technology
University of Turku
Joukahaisenkatu 3-5
20520 Turku
Finland

Reviewers

Professor Erkki Sutinen
School of Computing
University of Eastern Finland (Joensuu Campus)
P.O. Box 111, 80101 Joensuu
Finland

Professor Timo Honkela
Department of Contemporary Languages
University of Helsinki
P.O. Box 24, 00041 University of Helsinki
Finland

Opponent

Dr. Anna Korhonen
Department of Theoretical and Applied Linguistics
University of Cambridge Computer Laboratory
University of Cambridge
Faculty of English Building, 9 West Road
Cambridge CB3 9DB
United Kingdom

ISBN 978-952-12-3083-7
ISSN 1239-1883

Abstract

Much of the daily human communication uses the various natural languages of the world. This means that the amount of information available solely in textual or spoken form is tremendous. This motivates the development of different means to process such information automatically, in other words, applications in the field of *natural language processing* or *language technology*. The field encompasses a variety of topics and applications, from word counting and spellcheckers to syntactic parsing, and to advanced semantic applications such as information extraction, question answering, text generation and machine translation.

The state of the art in various tasks of natural language processing relies on statistical methods, which use a large amount of human-annotated examples to learn to analyze new examples. This is true of both *syntactic parsing*, which analyses the structures of sentences, and *semantic role labeling*, which concerns the argument structures of verbs, both important steps towards being able to process the meanings of natural language sentences.

Previously, no annotated text corpora for the purposes of syntactic parsing or semantic role labeling of Finnish have been available, which has been a severe hindrance to Finnish language technology research. This thesis addresses this important issue by presenting essential resources for Finnish: two human-annotated corpora containing annotation of both syntax and semantic roles. One corpus is from the domain of clinical Finnish, called *the clinical treebank and PropBank*, the other represents general Finnish and is known as *the Turku Dependency Treebank* and *the Finnish PropBank*. Additionally, the work presents two statistical parsers, one for clinical and one for general Finnish, induced from the annotated corpora. All of these represent the first freely available resources of their kind for the Finnish language.

As part of its main contribution, the work also studies the annotation process and the schemes used. It examines the question whether creating a custom annotation scheme from scratch can be avoided, seeing the substantial amount of work that such an endeavor would involve. In fact, the work shows that both resources can be created using existing annotation schemes for both syntax and semantic roles, with only minor modifications. This result is positive not only in reducing the work involved in the corpus

development, but also in increasing the compatibility of the resources with other, existing works created using the same scheme.

Further, the work explores different methods of annotation for these resources. It uses different annotation protocols for its different projects, always according to need, and shows that a high overall annotation quality can be achieved. It also presents a study which analyzes the difficulties of syntax annotation in particular: the most common errors made by humans and machines. The study shows that human annotators and statistical parsers share some difficulties and that a fairly frequent type of errors involves confusing two phenomena with each other. The same study presents a method for finding sentences most likely containing annotation errors so that these sentences can be subjected to further scrutiny to improve annotation quality in an efficient manner.

The resources presented in this thesis advance Finnish natural language processing by enabling a large amount of research that was previously impossible due to the lack of resources for the Finnish language. It is now attainable to study the statistical parsing of Finnish in detail and improve the accuracy of parsers, and similarly, this work also enables studies in Finnish semantic role labeling. Via these two applications, a path opens towards a large amount of semantic applications in understanding and generating language.

Acknowledgements

First and foremost, I would like to thank my supervisors Filip Ginter and Tapio Salakoski.

Filip, I don't even really know where to start. Thank you for your support and guidance, for all those countless times I showed at your door looking for answers and left having miraculously found them — if only to return with more questions moments later. Thank you for contributing an incredible amount of your time, effort, patience, mental health, blood, sweat and tears to our work that is now bearing fruit. Thank you for being there; even in times when I felt lost and confused, at least I wasn't lost alone.

Tapio, thank you for your always good advice, and for your help all those times it was so much needed. Most of all, thank you for believing in me, and believing in the work. Even with such an exotic topic (I'm told not all information technology grad students walk around the department carrying The Finnish Grammar), you continued to have faith in the project and support us in it.

My heartfelt gratitude also goes to our annotation team: Veronika Laippala, Timo Viljanen, Jenna Kanerva, Anna Missilä, and Stina Ojala. Without all of you, it's quite sure TDT wouldn't even exist, let alone be the great resource that it is today. On that same note, I would like to thank Samuel Kohonen, who solved numerous technical problems and squished many bugs before they had serious consequences. Thank you all for your tireless efforts on this humongous project of ours!

All this time, I've had the good fortune to be a part of a larger group of colleagues, who've given me not only help and advice, but also their friendly company. I would like to thank (there's no fair way to order you, so let me just resort to the alphabet) Antti Airola, Jari Björne, Kai Hakala, Juho Heimonen, Suwisa Kaephan, Juhani Luotolahti, Farokh Mehryary, Sebastian Okser, Tapio Pahikkala, Sampo Pyysalo, Olli Sjöblom and Hanna Suominen for their help and their company during the different phases of my PhD work. Outside the immediate group, I would like to thank the IKITIK consortium, especially Riitta Danielsson-Ojala, Heljä Lundgren-Laine, and Sanna Salanterä, as well as Lingsoft Ltd. and within, Juhani Reiman and Simo Vihjanen, for the fruitful collaboration.

This work wouldn't have been possible if it weren't for the help I've received from the staff of the Department of Information Technology and TUCS on more occasions than I can count. Thank you for making our daily lives as easy and pleasant as possible! Also when it comes to making this research possible, I'm grateful to the institutions that have provided funding for it: TUCS for its comittement to me as a doctoral student, and Turun Yliopistosäätio, TOP-säätiö, the Academy of Finland as well as Emil Aaltosen säätiö for the various grants awarded to me and for funding via projects I have worked for.

I'd like to take this opportunity to also thank professors Timo Honkela and Erkki Sutinen for their careful criticism of this thesis, and on a similar note, all the anonymous reviewers that have also offered their insights on the various papers that constitute the work of this thesis. I'm also very grateful to Dr. Anna Korhonen for agreeing to act as my opponent.

On a personal level, I'd like to thank my circle of friends. Throughout all of this time, before, during and after my doctoral studies, you've been nothing but kind and supportive. Thank you for your continued encouragement and for the many years together, of which I also hope there will be many to come. In particular, I would like to thank Antti Hakkala and Olli Heimo, who, perhaps unknowingly, have had a direct influence on this work. If it wasn't for Olli and his determination to help me find my place, I would never have met many of the people mentioned above, and indeed never have gotten started on making resources for Finnish. And if it wasn't for Antti and his tenacious peer support in my deepest hour of despair, I would never have finished.

Most importantly of all, my loving thanks go to my parents Marja and Seppo, who encouraged me to pursue an education in the first place and supported me all through these odd paths I've taken, and my sister Laura, who's simply been my best buddy for all these years. Without your constant, unwavering love and support, I don't know what I'd do.

All grammars leak.
— Edward Sapir

List of original publications included in the thesis

I Haverinen, K., Ginter, F., Laippala, V., and Salakoski, T. (2009). Parsing clinical Finnish: Experiments with rule-based and statistical dependency parsers. In Jokinen, K. and Bick, E., editors, *Proceedings of The 17th Nordic Conference on Computational Linguistics (NODALIDA'09)*, pages 65–72. Northern European Association for Language Technology (NEALT).

II Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., and Salakoski, T. (2010). Dependency-based PropBanking of clinical Finnish. In Xue, N. and Poesio, M., editors, *Proceedings of The Fourth Linguistic Annotation Workshop (LAW IV)*, pages 137–141. Association for Computational Linguistics (ACL).

III Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., and Salakoski, T. (2009). Dependency annotation of Wikipedia: First steps towards a Finnish treebank. In Passarotti, M., Przepiórkowski, A., Raynaud, S., and van Eynde, F., editors, *Proceedings of The Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 95–105. EDUCatt.

IV Haverinen, K., Viljanen, T., Laippala, V., Kohonen, S., Ginter, F., and Salakoski, T. (2010). Treebanking Finnish. In Dickinson, M., Müürisep, K., and Passarotti, M., editors, *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*, pages 79–90. Northern European Association for Language Technology (NEALT).

V Haverinen, K., Ginter, F., Laippala, V., Kohonen, S., Viljanen, T., Nyblom, J., and Salakoski, T. (2013). A dependency-based analysis of treebank annotation errors. In Gerdes, K., Hajičová, E., and Wanner, L., editors, *Computational Dependency Theory*, Frontiers in Artificial Intelligence and Applications, pages 47–61. IOS Press.

VI Haverinen, K., Nyblom, J., Viljanen, T., Laippala, V., Kohonen, S., Missilä, A., Ojala, S., Salakoski, T., and Ginter, F. (2013). Building the essential resources for Finnish: the Turku Dependency Treebank. *Language Resources and Evaluation*. In press. Available online. DOI: 10.1007/s10579-013-9244-1.

VII Haverinen, K., Laippala, V., Kohonen, S., Missilä, A., Nyblom, J., Ojala, S., Viljanen, T., Salakoski, T., and Ginter, F. (2013). Towards a dependency-based PropBank of general Finnish. In Oepen, S., Hagen, K., and Johannessen, J. B., editors, *Proceedings of The 19th Nordic Conference on Computational Linguistics (NODALIDA'13)*, pages 41–57. Linköping University Electronic Press.

List of related publications not included in the thesis

- Pyysalo, S., Ginter, F., Laippala, V., Haverinen, K., Heimonen, J., and Salakoski, T. (2007). On the unification of syntactic annotations under the Stanford Dependency scheme: A case study on BioInfer and GENIA. In Cohen, K. B., Demner-Fushman, D., Friedman, C., Hirschman, L., and Pestian, J., editors, *Proceedings of Biological, translational and clinical language processing (BioNLP'07)*, pages 25–32. Association for Computational Linguistics (ACL).

- Haverinen, K., Ginter, F., Pyysalo, S., and Salakoski, T. (2008). Accurate conversion of dependency parses: Targeting the Stanford scheme. In Salakoski, T., Rebholz-Schuhmann, D., and Pyysalo, S., editors, *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), Turku, Finland*, pages 133–136. Turku Centre for Computer Science (TUCS).

- Haverinen, K., Ginter, F., Laippala, V., Kohonen, S., Viljanen, T., Nyblom, J., and Salakoski, T. (2011). A dependency-based analysis of treebank annotation errors. In Gerdes, K., Hajičová, E., and Wanner, L., editors, *Proceedings of The First International Conference on Dependency Linguistics (Depling'11)*, pages 115–124. Pompeu Fabra University.

- Haverinen, K. (2012). Syntax annotation guidelines for the Turku Dependency Treebank. Technical Report 1034, Turku Centre for Computer Science. Second edition, Revised for the treebank release of July 2013.

- Nyblom, J., Kohonen, S., Haverinen, K., Salakoski, T., and Ginter, F. (2013). Predicting conjunct propagation and other extended Stanford Dependencies. In Hajičová, E., Gerdes, K., and Wanner, L., editors, *Proceedings of The Second International Conference on Dependency*

Linguistics (Depling'13), pages 252–261. Charles University in Prague, Matfyzpress.

- Ginter, F., Nyblom, J., Laippala, V., Kohonen, S., Haverinen, K., Vihjanen, S., and Salakoski, T. (2013). Building a large automatically parsed corpus of Finnish. In Oepen, S., Hagen, K., and Johannessen, J. B., editors, *Proceedings of The 19th Nordic Conference on Computational Linguistics (NODALIDA'13)*, pages 291–300. Linköping University Electronic Press.

- Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., Manning, C. D. (2014). Universal Stanford Dependencies: a cross-linguistic typology. In Calzolari, N., Choukri, K., Declerck, T., Loftsson, H., Mariani, J., Moreno, A., Odijik, J., and Piperidis, S., editors, *Proceedings of The Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 4585–4592. European Language Resources Association (ELRA).

Contents

Chapter 1

Introduction

This thesis describes research in the field of *natural language processing* or *computational linguistics*, or more precisely, in the area of corpus development. The work is organized as follows. First, we briefly introduce the field of natural language processing as well as the topics of syntactic parsing, treebanks, and semantic role labeling, all closely related to the topic of the thesis. We also describe related work, especially on the Finnish language, and finally, we end the introduction by stating the goals of the research presented in this thesis. In the second chapter, we describe two treebanks of Finnish: the clinical Finnish treebank and the Turku Dependency Treebank. We discuss the text sources as well as the annotation methods of the two treebanks, and also present quality evaluations. In this context, we also discuss parsing experiments based on these corpora, and present a study of common annotation errors. The third chapter introduces a further level of analysis for both of these treebanks: the PropBank annotation. Finally, we conclude the work and describe future work directions.

1.1 Natural language processing

Natural language processing is an interdisciplinary field concerning both human language and technology, or, as defined by Jurafsky and Martin (2009, p. 1), a field whose aim is to "get computers to perform useful tasks involving human language, tasks like enabling human–machine communication, improving human–human communication, or simply doing useful processing of text or speech". The field encompasses a large variety of different tasks on different levels of language: from simple word-counting or sentence-splitting applications to *information extraction*, which aims to capture facts from running text, *machine translation*, where text is automatically translated from one language to another, and *question answering*, where a software answers questions formulated by the user.

NLP applications generally aim for either *understanding* natural language on some level, in which case their task is called *analysis*, or *producing* it, in which case it is called *generation*. Most applications are also aimed at either *text* or *speech*. This thesis in concerned with the *analysis* of language in *text* form.

In order to achieve the ambitious goal of analyzing text, the task is usually separated into several subtasks that can each be attended to separately. A frequent, even standard, division is that of *sentence-splitting*, *tokenization*, *morphological analysis*, *syntactic analysis* and *semantic analysis*. These subtasks are often performed sequentially, each step using information provided by the previous step. However, it is also possible to combine some of these steps into one processing phase, in a technique termed *joint inference*. This strategy has been used for instance in the studies by Bohnet and Nivre (2012) and Bohnet et al. (2013).

Sentence-splitting is the task of determining how a given text should be divided into sentences and similarly, *tokenization* divides sentences into words. *Morphological analysis* concerns with the structures of individual words and aims to present all components that a word consists of. For instance, a morphological analysis of the English wordform *dogs* would reveal that it is a plural form of the noun *dog*, which consists of the *baseform* or *lemma dog* and the plural suffix *s*. *Syntax* is the study of the sentence structure, that is, the way in which sentences are constructed from their individual words. *Semantics* is the study of meaning, and it can be further subdivided into *lexical semantics*, which studies word meanings, and *sentence semantics*.

This thesis considers two subtasks in the analysis sequence in particular: automatic syntactic analysis or *parsing* and one particular type of semantic analysis, that of *semantic role labeling*.

1.2 Parsing and treebanks

The task of *parsing* is that of automatic syntactic analysis, or, in other words, automatically determining the syntactic structure of a sentence. Nivre (2006) distinguishes two kinds of parsing: *grammar parsing*, where parsing is in relation to some formal grammar and the task is to assign a given sentence all analyses allowed by the grammar, and *text parsing*, where no formal grammar is assumed and a single analysis of each sentence is aimed at. This thesis is concerned with the latter, for which Nivre states the problem of parsing as:

> Given a text $T = (\mathrm{x}_1,...,\mathrm{x}_n)$ in the language L, derive the correct analysis for every sentence $\mathrm{x}_i \in T$.

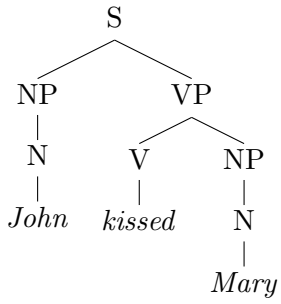

Figure 1.1: A constituency analysis of the sentence *John kissed Mary*.

In both parsing and general linguistics, there are two broad ways to represent the syntax of a sentence. In *constituency syntax*, sentences consist of *constituents* or *phrases*, which in turn may recursively contain more constituents. A *constituent* is a string of words that can act as a single unit, much like a word. For example, the string *that small boy* is a constituent: it could be replaced with the single word *he* in a sentence, without the replacement affecting the grammaticality of the sentence. Each constituent also has a type, such as a *noun phrase* or a *verb phrase*. A constituency analysis of a simple English sentence is given in Figure 1.1. Constituency grammar originates from the early work of Bloomfield (1933), and perhaps the most influential work on the topic is that of Chomsky (1957), on which many of the contemporary theories are based.

Dependency syntax, in turn, relies on the concept of *dependencies* between individual words of the sentence. Typically, the dependencies are binary, directed and labeled. The *direction* indicates that one of the words is the *head* or *governor* and the other its *dependent*. It is also sometimes said that a word *governs* another, or in the opposite direction, a word *depends* on another. The *label* or *type* of the dependency describes the syntactic function of the dependent word. The best-known proponents of dependency syntax are likely Tesniére (1959) and Mel'čuk (1979, 1988). A dependency analysis for the example sentence of Figure 1.1 is given in Figure 1.2.

Some dependency theories, as for instance those of Mel'čuk (1979, 1988), require all dependencies of a representation to be strictly syntactic in order to accept it as a dependency representation. However, especially in the context of computational linguistics and natural language processing, the term *dependency* is usually understood broadly: as simply a binary relation between words. This thesis also adopts a broad view of dependencies, considering schemes such as the Stanford Dependency Scheme (SD) (de Marneffe and Manning, 2008a,b; de Marneffe et al., 2013, 2014), Grammatical Relations (GR) (Carroll et al., 1999) and Link Grammar (Sleator and Temperley, 1993) dependency representations.

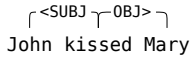

Figure 1.2: A dependency analysis of the sentence *John kissed Mary*.

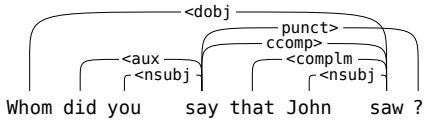

Figure 1.3: A dependency analysis of the non-projective sentence *Whom did you say that John saw?* When drawn above the words of the sentence, the dependencies cross.

Dependency formalisms impose restrictions on the structures they consider valid. Commonly, an analysis is considered valid only if the structure formed by the dependencies is a *tree*, that is, if the sentence has one *root* token that directly or indirectly governs all other tokens and each token only has one governing token, not several. Some formalisms only allow *projective* trees: informally, trees where, when drawn above the sentence as graph edges, no dependencies cross and where the root token is not covered by a dependency. Also from the point of view of parsing, some parsers are only able to produce projective structures. However, this restriction reduces the expressive power of a dependency formalism, as especially so called *free word order languages* in fact contain non-projective structures. One non-projective structure in English is illustrated in Figure 1.3.

Both constituency and dependency formalisms are commonly used within natural language processing, but in recent years, dependency representations have grown increasingly popular, and they have been argued to be preferable over constituency ones. For instance, Lin (1998) has proposed that dependency representations should be used in the evaluation of parsers, even those parsers that natively produce a constituency output. Clegg and Shepherd (2007), in turn, have argued that dependency representations are beneficial in applications utilizing parser output, and indeed, dependencies have been successfully used in application-oriented contexts. For instance, the CoNLL'08 (Surdeanu et al., 2008) and CoNLL'09 (Hajič et al., 2009) shared tasks evaluated a number of systems performing the task of semantic role labeling (see Section 1.3) on top of dependency parsing. For more examples of applications that utilize dependency parsing, see Section 2.1, which discusses the representation used in this work, the Stanford Dependency scheme.

The methods used for parsing are divided into two main approaches, similarly to many other NLP tasks. *Rule-based* parsers are developed by manually encoding the grammar of the desired language into rules, whereas

statistical parsing systems utilize a text collection with pre-built syntactic structures to automatically induce an analyzer of the language using statistical methods.

There are several benefits to the latter method. First, while both approaches require manual effort — in rule-based parsing, the effort is in creating the grammar rules, and in statistical parsing, the pre-built syntactic structures — the statistical method has the benefit that the manual effort is only needed once, after which statistical methods can be applied to the data multiple times, to induce different parsers. Once a sufficiently large body of training data exists, the parser performance can be improved by improving the statistical methods. Second, the methods used to induce a parser are for the most part language independent; once a body of manually created examples is completed for a new language, existing methods can be used to induce a parser for this language with little effort. Finally, due to the small amount of manual effort involved in statistical parser induction itself, statistical parsers are often freely available to the research community.

The manually created resource for parsing, or any NLP task, is called an *annotated corpus*, and it is typically used for a *supervised* machine learning approach, meaning that the human-made markings are used to train the method. The term *corpus* refers to an electronic collection of text, and *to annotate* is to manually mark the desired properties of text into the corpus. Different purposes call for different annotations; common annotations in corpora include annotation of parts-of-speech (POS), syntactic structures of sentences and semantic relations. A text corpus with *syntax* annotations is called a *treebank*.

An annotated corpus can be used for two main purposes. First, it serves as training material for a statistical method, and second, it can be used for evaluating the method. The evaluation is done by using the corpus as a so called *gold standard* against which the output of the method can be compared. In the latter purpose, the utility of a corpus does not depend on the method being statistical, but also rule-based methods can and should be evaluated against a gold standard.

Due to the benefits of statistical methods and annotated data outlined above, this thesis concerns with the statistical approach to parsing, and its main focus is on creating a treebank for one particular language, Finnish. Many treebanks for the different languages of the world have been created, the best-known of them being perhaps the Penn Treebank (Marcus et al., 1993) for English and the Prague Dependency Treebank (Hajič, 1998) for Czech. Other languages with an existing treebank include for instance Swedish, German, Estonian, Japanese, Arabic, Chinese, and even dead languages, such as Latin and Ancient Greek. However, prior to the beginning of the work presented in this thesis, there was no publicly available Finnish treebank of sufficient size for statistical parsing, which has been a major hin-

drance for Finnish NLP. One of the biggest goals of this work is to remove this obstacle by providing such an essential resource for Finnish.

1.3 Semantic role labeling and related resources

A syntactic parse tree, despite its immense usefulness, by no means provides a full analysis of the meaning of the sentence. As noted by Palmer et al. (2010), it does not specify "Who did What to Whom, and How, When and Where", that is, it does not give an explicit account of the events and participants present in the sentence. This additional information is given by the analysis of *argument structures of verbs* or *semantic roles*. Similarly to syntactic analysis, the state of the art in *semantic role labeling (SRL)* is based on statistical methods that learn the roles from a manually annotated corpus.

To demonstrate the necessity of defining semantic roles in addition to syntactic functions, consider the following two simple English examples.

Example 1 *John opened the door.*

Example 2 *The door opened.*

Syntactically, the *door* is the direct object in Example 1, but the subject in Example 2. However, it still represents the same participant (the thing opening) in both sentences, despite the different syntactic means of expressing this fact. In other words, the *door* occupies the same *semantic role* in both example sentences. It should also be noted that both of these examples are in the active voice, that is, neither is a passivized version of the other. Passivization (as in *The door was opened by John*) would provide syntactic evidence for the fact that the *door* occupies the same role in both sentences, but no such evidence is present here.

The questions of how many different semantic roles exist and what kinds of roles they are have proved difficult, and consequently, there are many different ways to analyze them. Table 1.1 presents a listing of roles that are mostly generally agreed upon. As a manifestation of the possible different ways of analysis, several projects on semantic roles targeted at the English language have taken place. One of the best-known ones is the *FrameNet* project (Baker et al., 1998), which aims for very fine-grained labels for groups of different verbs. For instance, verbs of cooking receive roles such as *Food* and *Cook*. Another widely known effort, the *VerbNet* project (Dang et al., 1998) defines *classes* of verbs and roles for them. VerbNet defines 24 different role labels, akin to those presented in Table 1.1. Finally, The *Proposition Bank* or *PropBank* project (Palmer et al., 2005) uses the most coarse-grained labels, and defines them on a verb-by-verb basis. Out of the three, PropBank

Role	Description
Agent	Initiator of action, volitional
Patient	Affected by action, undergoes state change
Theme	Entity moving, or being "located"
Experiencer	Experiences event, not in control
Beneficiary	Entity for whose benefit the action is
Instrument	Intermediary or means
Location	Place of object/action
Source	Starting point
Goal	Ending point

Table 1.1: The most generally agreed upon semantic roles. Table adapted from Palmer et al. (2010).

is the only resource intended as a large annotated text corpus rather than a lexicon of verbs.

The work of this thesis applies the PropBank annotation scheme to Finnish. This particular scheme is further described and its use motivated in Section 3.1. As was the case with the treebank, no publicly available SRL resource for Finnish existed prior to this work, making it difficult to do research on semantic role labeling of Finnish. Removing this obstacle is the second part of the main contribution of this thesis.

1.4 Related work

This Section discusses previous and concurrent work in the field of automatic syntactic and morphological analysis with respect to the Finnish language. The main focus is on tools and resources that have been utilized in the work of this thesis.

In the area of full syntactic analysis, tools and resources prior to this work are rather scarce. The only previously existing full syntactic parser of Finnish is the Connexor Machinese Syntax parser,[1] which is a closed source commercial product. The lack of a freely available statistical parser was one of the most important motivations for building the Turku Dependency Treebank, the early versions of which constitute the first publicly available treebank of general Finnish. Shortly after the second release of Turku Dependency Treebank, a treebank consisting of the grammar examples from the Finnish reference grammar (Hakulinen et al., 2004), FinnTreeBank (Voutilainen and Lindén, 2011) was made available. This treebank has been built for the purposes of rule-based parsing and is intended to be, as Voutilainen and Lindén (2011) put it, a "grammar definition corpus".

[1] `http://www.connexor.eu`

FinnTreeBank has since its original release been augmented with small samples from three other genres: Wikipedia, news text from *Helsingin Sanomat* and *Tietoviikko*, and a sample from Jostein Gaarder's fictional work, *Sophie's World*. The grand majority of FinnTreeBank (approximately 97%), however, consists of grammar examples even in the augmented version.

In Paper VI, we use a sample of the grammar examples from FinnTreeBank as one section of TDT. The purpose of this, in addition to gaining text from this genre, is to enable a conversion between the schemes of the treebanks, which was required for the purposes of a parsebank of Finnish, as will be briefly described in Section 2.3.7.

In the area of semantic role labeling, no tools and resources for Finnish existed prior to the work of this thesis. This is rather understandable, seeing that semantic role labeling generally depends on parsing, for which resources were previously also inadequate.

In morphology tools, however, the situation is better. The best-known morphological analyzers for Finnish are likely FinTWOL and FinCG (Karlsson, 1990; Koskenniemi, 1983) by Lingsoft Inc., a morphological analyzer and a constraint grammar-based morphological disambiguator, respectively. Both of these analyzers are closed source commercial tools. In addition, an early work called the Kielikone parser by Valkonen et al. (1987) resolves morphological ambiguity and produces elementary syntactic functions. During the work described in this thesis, two open source morphological analyzers, OMorFi (Pirinen, 2008; Lindén et al., 2009) and Voikko,[2] have become available for Finnish. Out of these two, OMorFi has the broader vocabulary coverage, and Voikko is used primarily in open source spellchecker applications for Finnish. OMorFi and Voikko, like FinTWOL, produce all possible analyses for a given word with no context-based disambiguation, whereas FinCG and the Kielikone parser are intended for deciding between readings. In addition to the above, also for instance Silfverberg and Lindén (2010) have worked on POS-tagging of Finnish, using weighted finite state transducers.

Partly due to the time when the open source morphological tools for Finnish emerged, the works described in this thesis use both FinTWOL and OMorFi for retrieving morphological analyses of words. The clinical treebank and PropBank described in Papers I and II, as well as earlier versions of the Turku Dependency Treebank (see Papers III, IV and V) are based on analyses produced by FinTWOL and disambiguated by FinCG, whereas the latest release of the Turku Dependency Treebank, as described in Paper VII, contains morphological analyses based on readings given by OMorFi. The reason for changing the tool was our desire to make both the

[2] `http://voikko.sourceforge.net/`

treebank and the accompanying parsing pipeline with all of its components freely available under an open licence. This desire could only be completely fulfilled once an open source analyzer for morphology was available.

Overall, the resources prior to this work are the following. Morphology tools were already in existence, including an open source analyzer that became available during this work. Syntactic resources, however, were scarce, including only one full parser of Finnish, one that is a closed source commercial product. In particular, no freely available large-scale treebank or large-coverage statistical parser were available. Similarly, no SRL resources or tools for Finnish were available. These two lacks in resources have been of severe hindrance to Finnish NLP, already in themselves and also due to several advanced language technology applications depending on parsing, semantic role labeling, or both. This thesis aims to solve this problem, as will be detailed in the next subsection.

1.5 Objectives of the work

The overall research objective of this thesis is to enable and advance Finnish natural language processing research by creating the resources so far not available to the community, and to answer the question *how* such resources can be constructed efficiently, yet so that the quality is as high as possible.

This objective can be divided into four subgoals, which belong to the areas of treebanks and parsing, semantic role labeling resources, annotation schemes and the annotation process. Each subgoal is examined from two perspectives: those of clinical and general Finnish. The work on clinical Finnish has, in addition to its independent purpose as part of a clinical language processing project, been used as a pilot project for the larger work on general Finnish. It has been ideal for this purpose, seeing that the corpus produced is clearly smaller in scale, yet it enables one to make some predictions on whether a similar but larger undertaking would be feasible and if so, how it should be conducted. The following describes the four subgoals in more detail.

Treebanks and parsing The first and foremost research objective of this work is to create the resources previously missing for Finnish: a freely available treebank suitable for statistical parsing, and a first statistical parser. Paper I addresses this part of the objectives for clinical Finnish by introducing the *clinical treebank*, while Papers III, IV, V and VI concentrate on these issues in the general domain and present the *Turku Dependency Treebank*. These two resources constitute the first freely available treebanks of clinical and general Finnish, and enable a large amount of Finnish NLP.

Semantic role labeling resources The second objective of the work is to further advance Finnish NLP by creating resources for semantic role labeling. The first two objectives are tightly coupled, due to the semantic role labeling resources being built on top of TDT and the clinical treebank, as is usually the case with these kinds of resources. In this sense, the second objective extends on the first. Again, this second objective is addressed in two parts: Paper II describes work done in the clinical domain, and Paper VII is concerned with the creation of a semantic role labeling resource in the general domain.

Annotation schemes The third part of the objectives is to find a suitable scheme for annotating Finnish for syntax on the one hand, and semantic roles on the other hand. This raises questions on whether building a custom scheme for one or both of these purposes can be avoided, and if so, what modifications would be required to the existing scheme used instead. If possible, using an existing scheme would have benefits in for instance compatibility with other resources, but whether such a scheme suits Finnish is not knowable *a priori*. Papers I through VII all in part contribute towards answers to these questions. Paper V addresses this sub-objective by examining the difficulties faced by annotators in the syntax phase with the particular scheme used.

The annotation process The fourth and final objective of this work is to find an annotation workflow or process that suits the needs of these annotation projects. Slightly different approaches have been used in the different studies throughout this thesis, each deemed suitable for the current purpose. Paper V studies the annotation process itself and the errors produced by the annotators, suggesting ways in which annotation could be made more fluent.

The contributions of all papers included in the thesis, as well as some other related papers are summarized in the following, in rough chronological order. This serves as an illustration of the individual papers of this work, as well as their relations to each other and the objectives outlined above. The relationships of the papers and their topics are also illustrated in Table 1.2.

Pyysalo et al. (2007) *On the unification of syntactic annotations under the Stanford Dependency scheme: a case study on BioInfer and Genia* was the first work in the group to use the SD scheme. It converted the existing annotation of a biomedical English corpus to SD, using it as an unifying scheme for parser comparison purposes.

Haverinen et al. (2008) *Accurate Conversion of Dependency Parses: Targeting the Stanford Scheme* also used SD for English; it converted the

	Paper I	Paper II	Paper III	Paper IV	Paper V	Paper VI	Paper VII
Clinical treebank	✓						
Turku Dependency Treebank			✓	✓	✓	✓	
Clinical PropBank		✓					
Finnish PropBank							✓
Applying the SD scheme	✓		✓		(✓)	✓	
Applying the PB scheme		✓					✓
Annotation process	(✓)	(✓)	✓	✓	✓	✓	✓
Parsing	✓			(✓)	(✓)	✓	

Table 1.2: The papers and topics of the thesis.

output of a biomedical parser to SD to be able to compare its performance against other parsers.

Paper I (2009) *Parsing Clinical Finnish: Experiments with Rule-Based and Statistical Dependency Parsers* first applied the SD scheme to Finnish. It presented the clinical treebank as well as parsing experiments on clinical Finnish.

Paper III (2009) *Dependency Annotation of Wikipedia: First Steps towards a Finnish Treebank* described the need for a freely available treebank of Finnish and presented the first release of TDT. It begun the annotation effort using Wikipedia texts and was the first work to apply the SD scheme to general Finnish.

Paper II (2010) *Dependency-based PropBanking of Clinical Finnish* first applied the PropBank scheme to Finnish. It extended the clinical treebank in size and annotation scope as well as presented the clinical PropBank.

Paper IV (2010) *Treebanking Finnish* extended TDT from Paper III and reported first results on annotator accuracy. It also studied the effect of pre-annotation on annotator accuracy and speed.

Paper V (2011, extended in 2013) *A Dependency-based Analysis of Treebank Annotation Errors* studied the typical annotation errors of humans and parsers as well as presented a method for recognizing likely annotation errors. It also extended TDT again.

Paper VI (2013) *Building the Essential Resources for Finnish: the Turku Dependency Treebank* presented the ultimate release of TDT and dis-

cussed the full selection of text sources and the final annotation scheme. It also presented annotation statistics as well as a baseline parser for Finnish.

Haverinen (2012) (second edition in 2013) *Syntax Annotation Guidelines for the Turku Dependency Treebank* discussed the details of the final annotation scheme of TDT.

Paper VII (2013) *Towards a Dependency-based PropBank of General Finnish* presented the early stages of the Finnish PropBank. It was the first work to apply the PropBank scheme to general Finnish and discussed the Finnish-specific issues in creating this resource. In addition, it presented preliminary annotator accuracy figures.

Nyblom et al. (2013) *Predicting Conjunct Propagation and Other Extended Stanford Dependencies* presented a method for predicting additional SD dependencies (see Section 2.3.3), using TDT as training data.

Ginter et al. (2013) *Building a Large Automatically Parsed Corpus of Finnish* used TDT and a parser induced from it to build a parsebank with the Europarl and JRC-Acquis corpora as text sources.

de Marneffe et al. (2014) *Universal Stanford Dependencies: a Cross-linguistic Typology* presented a new version of the SD scheme, intended to be universally applicable across languages. This new scheme version adopts ideas and relations from treebanks based on SD, including TDT.

Chapter 2

Treebanks: clinical and general domain corpora of Finnish

As discussed in the previous Chapter, the biggest contribution of this thesis are two annotated corpora of Finnish: one consisting of texts from the clinical domain, and one of different text sources in the general domain. This Chapter describes the text corpora as well as the syntactic aspects of the work, the treebanks.

We begin by discussing the Stanford Dependency scheme, the syntactic annotation scheme used in the corpora of this work. We then move on to the clinical domain corpus described in Paper I (and in some respects also in Paper II), and after that, to the general language treebank known as the Turku Dependency Treebank or TDT, which has been described in Papers III, IV, V and VI, as well as the technical report by Haverinen (2012). Section 2.3.7 describes the baseline parser induced from TDT and released as part of the work of Paper VI, and finally, in Section 2.4 we discuss common annotation errors and their detection, which have been studied in Paper V.

2.1 The Stanford Dependency (SD) scheme

The syntactic representation used in the work of this thesis, the *Stanford Dependency*[1] or *SD* scheme (de Marneffe and Manning, 2008a,b; de Marneffe et al., 2013, 2014) has recently become popular among dependency representations used for parsers, treebanks and practical applications. The scheme was originally designed for English (see Figure 2.1 for an illustration of the

[1] called either *Stanford Dependency*, as in this thesis, *Stanford Dependencies*, or *Stanford typed Dependencies*

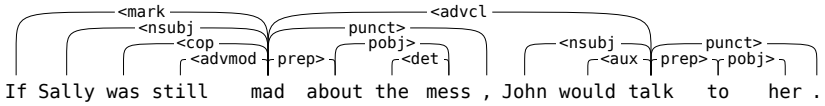

Figure 2.1: The original SD scheme on English.

English scheme), but one of its design principles is language independence. This is demonstrated by a recent multilingual effort by McDonald et al. (2013) that has produced a collection of, in its recently released second version, eleven different treebanks, including also TDT. All of these treebanks use (either natively or by conversion) a unified syntax annotation scheme, which is a minor modification of the SD scheme.

We have chosen the SD scheme to be used in both of the treebanks in this thesis based on several of its properties. One of them is the aforementioned language independence, as naturally, we wished to have an existing scheme that could be applied to Finnish with as few modifications as possible. In addition, the scheme is intended for practical applications, which is also desirable for the treebanks considered in this work, seeing that their primary motivation lies in parsing and further NLP applications. One part of this practical orientation is that SD is designed to be readable also for the developers and users of applications, that is, people who do not necessarily have a background in linguistics. An additional benefit from a practical, application-oriented point of view is having a common annotation scheme with other resources, such as the previously mentioned treebank collection of McDonald et al. For instance machine translation applications can benefit from multilingual resources with a common syntax representation. Finally, we had previous experience of the scheme on the English language, in the biomedical domain, from developing the syntax annotation of the BioInfer corpus (Pyysalo et al., 2007) as well as creating an annotation scheme conversion (Haverinen et al., 2008) from the scheme of the Pro3Gres parser (Schneider et al., 2004) into the SD scheme.

Other treebanks natively annotated using the SD scheme include a treebank for Chinese (Lee and Kong, 2012) and one for Persian (Seraji et al., 2012), and via the collection of McDonald et al. (2013), SD treebanks for Brazilian Portuguese, English, French, German, Italian, Indonesian, Japanese, Korean, Spanish and Swedish now also exist. Further, any (English) treebank annotated in the constituency scheme used in the Penn Treebank (Marcus et al., 1993) can be converted into the SD scheme using a conversion provided in the original Stanford tools.[2] In addition to the Stanford parser (Klein and Manning, 2003), several other parsers are also able to produce output in the SD scheme, such as the Clear parser (Choi and

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

Palmer, 2011) and the parser of Tratz and Hovy (2011). Quite naturally, also any dependency parser that can be trained using a treebank can produce this scheme, including for instance the MaltParser (Nivre et al., 2007), the MSTParser (McDonald et al., 2006) and the MateTools parser (Bohnet, 2010). The scheme has been popular in parser evaluation works in particular (Cer et al., 2010; Nivre et al., 2010; Miwa et al., 2010; Foster et al., 2011; Clegg and Shepherd, 2007), and it has also been successfully used in applications utilizing parser output (Björne et al., 2010; Miyao et al., 2009; Qian and Zhou, 2012; Zhuang et al., 2006; Meena and Prabhakar, 2007).

2.2 The clinical treebank

In chronological order, the first annotated corpus of those presented in this thesis is a treebank consisting of clinical Finnish, henceforth referred to as *the clinical treebank*. We begin by describing the text of the treebank and its particular language. Next, we discuss the syntax annotation and the annotation process of the treebank, and finally, parsing clinical Finnish.

2.2.1 ICU Finnish

The motivation for creating a clinical domain treebank for Finnish stems from the needs of the IKITIK consortium,[3] or its early incarnations in the year 2009. This consortium aims to develop language technology solutions as well as other means to process clinical information for health care. The overall aim is to aid both health care professionals and the people using the services by improving the clarity, understandability and accessibility of patient documents. One step in this direction is naturally the automatic syntactic analysis of these documents.

The clinical treebank in its final version, as described in Paper II, contains 15,335 tokens (2,081 sentences)[4] of a specific language, termed *ICU (Intensive Care Unit) Finnish*. The contents of the corpus come from patient reports written by nurses in the intensive care unit of a Finnish hospital. These reports describe the condition of a single patient and its development over time; they are written by the nurse during a work shift. Often these reports cover certain standard topics, such as oxidation, diuresis, or relatives.

ICU Finnish has some defining characteristics that set it apart from standard Finnish, and some of these characteristics also present challenges for its analysis. The sentences of ICU Finnish are typically short, and it may be difficult to establish sentence boundaries, as it is common to bind together simple main clauses using merely a comma, or in some cases, no surface

[3] `http://www.ikitik.fi/en/`

[4] Paper I describes an earlier subset of 1,019 sentences with 7,614 tokens.

Potilas melko väsynyt.	Patient fairly tired.
Hapettuu hyvin.	Oxidates well.
DIUREESI: riitävä	DIURESIS: suficient
Annettu 300mg [lääke], vaste hyvä, jaketaan illalla	Given 300mg [drugname], response good, will cotinue in the evening
OMAISET: sisko soittanut, vaimo käymässä	RELATIVES: sister called, wife visited

Figure 2.2: Examples of ICU Finnish (left column) and their exact translations (right column), preserving typical features such as spelling errors and capitalization.

marking at all. Spelling errors are frequent due to the nature of the work in an intensive care unit: little time is left for documentation. Also technical terms occur frequently and present a problem for morphological analysis, as do the spelling errors. The style of the patient reports is telegraphic, and different sentence elements are often omitted, as they can be inferred from the context. Figure 2.2 gives an illustration of the specific features of ICU Finnish.[5]

2.2.2 Applying the SD scheme to ICU Finnish

The work of Paper I is the first of our works to apply the SD scheme to Finnish. Partly due to the nature of Finnish, and partly to the specific characteristics of ICU Finnish in particular, the scheme used in this annotation project differs in some respects from the original SD scheme. This scheme version is an early incarnation of the Finnish-specific SD scheme used later for the Turku Dependency Treebank.

Overall, the SD scheme was found to suit Finnish well, and the minor adjustments made to close the remaining gaps are illustrated in Figure 2.3. One of these adjustments concerned the treatment of adpositional phrases. In the English SD scheme, the preposition is the head of a prepositional phrase. This was changed in order to accommodate the typical Finnish phenomenon where an inflected nominal modifier with no adposition is used, and to analyze it similarly to adpositional phrases. Also, as Finnish does not have passive subjects, what would be the passive subject in the English SD scheme is instead analyzed as the direct object in the ICU Finnish version of the scheme. Both of these changes were later adopted in the general Finnish version of the scheme in TDT.

[5]Due to the confidential nature of patient reports, this and all other examples in this thesis as well as in Papers I and II are artificial rather than genuine examples from the data.

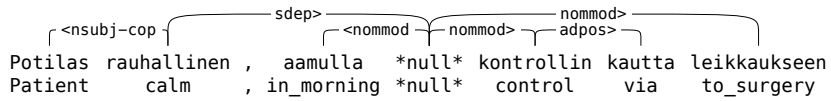

Figure 2.3: The SD scheme on ICU Finnish. The *sdep* dependencies join together loosely connected clauses and the *null verb* stands for an omitted main verb. Note also the new dependency types *nommod*, *adpos* and *nsubj-cop*, which later became part of the general Finnish SD scheme. The example sentence can be translated as *Patient calm, in the morning to the surgery via control*.

The nature of the ICU language, in turn, also required adaptations for its specific features. As described above, establishing sentence boundaries in this language is less than straightforward, seeing that commas are often used instead of periods, or no surface marking at all is used between sentences. Our solution to this problem is to only split the text into different sentences where the boundary is clearly surface-marked, and use a specific dependency type *sdep* to mark clauses joined together loosely.

A very frequent phenomenon in ICU Finnish is that of omitting sentence elements. For instance, copulas and auxiliaries or even the main verbs of sentences are often left out, as they can be inferred from the surrounding context, as in *Potilas leikkaukseen (Patient to surgery)*. A missing main verb would be a problem for any dependency scheme, seeing that the main verb or predicate is considered the head word of the whole sentence. We have solved this issue by introducing an extra token to those sentences missing the main predicate (see Figure 2.3 for an illustration). In Papers I and II these tokens are termed *null verbs* due to the fact that in the clinical treebank they represent missing main verbs. These extra tokens return in TDT, termed *null tokens*, as in general Finnish they can also represent parts-of-speech other than verbs, although verbs are the most common case. In both corpora, null tokens are used only when they are necessary in order to construct an analysis. That is, no null token is inserted for e.g. copulas and auxiliaries, as in the SD scheme, a copula or an auxiliary does not act as the head of a clause, but rather depends on the main verb or predicative. Further, these verbs do not have dependents that would be present when the verbs themselves are absent.

As a related, small adjustment, we have changed the dependency type for nominal subjects in copular contexts from *nsubj* to *nsubj-cop*, to signal that the governor of this dependency is not expected to be a verb, as in the regular case of subjects, but for instance a noun or an adjective. This is especially important with the frequent omissions of copulas in the ICU language, as it is helpful in locating copular structures even if the copula itself is absent. This change, too, later migrated to the general Finnish

version of the scheme, where also clausal copular subjects received the same treatment.

The clinical treebank has been annotated using *full double annotation*. This means that each sentence is given to two different annotators, who first independently annotate them, and afterwards any disagreements are solved together with all annotators. This is a simplified version of the annotation protocol of TDT, which is detailed in Section 2.3.5.

2.2.3 Parsing experiments

Despite the relatively small size of the clinical treebank (at the time of the writing of Paper I, 1019 sentences with 7614 tokens), it nevertheless enabled parsing experiments on ICU Finnish. The experiments of Paper I include the comparison of two parsers: a previously existing rule-based parser by Laippala et al. (2009) that has been specifically developed for the ICU language, and a statistical parser induced from the clinical treebank using the MaltParser (Nivre et al., 2007) system.

The parser of Laippala et al. produces output in a constituency-based scheme, meaning that in order to compare its performance against that of MaltParser, a scheme conversion was necessary. This conversion was implemented first using the head words of each phrase as assigned by the parser of Laippala et al. to build the dependency structure of the sentence and then assigning the dependency types using a set of hand-written rules. In few exception cases, the rules were required to change the existing dependency structure. This conversion is similar to that of the original Stanford tools,[6] which is able to transform trees in the Penn Treebank constituency scheme into SD analyses.

We measured the parsing performance using *labeled attachment score (LAS)*, which is defined as the proportion of tokens, out of all tokens, that receive the correct head word and the correct dependency type. In the comparison, the rule-based parser achieved an overall performance of 75.2% in LAS, whereas the performance of the statistical parser was 69.9% LAS, that is, slightly lower in absolute terms. However, two things should be noted about these results. First, the rule-based parser does not have a ranking component, which means that it produces several parse trees for each sentence (33 on average) and is unable to select between them. Second, the coverage of the rule-based parser is approximately 75%; that is, 75% of the sentences in the corpus receive at least one analysis. Due to these limitations, the performance of the rule-based parser was measured so that for each sentence, the best parse of all the alternatives produced was evaluated (*oracle best parse performance*), and the evaluation was limited to only those sentences that received at least one analysis. Therefore, the performance

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

should be considered an upper bound for the rule-based parser. Taking into account the rather small size of the corpus used for training the statistical parser, it seems likely that with additional training data (later provided in the work of Paper II), the statistical approach would in fact be preferable over the rule-based parser of Laippala et al.

This tentative finding of Paper I has later been confirmed. Laippala et al. (2013) have further studied the statistical parsing of different varieties of clinical Finnish, including ICU Finnish. By further extending the clinical treebank as well as using training data from other clinical sources they have achieved a LAS of 84.6% for ICU Finnish.

The parsing results in Paper I as well as the general result that an ICU Finnish treebank was possible to create using the SD scheme encouraged us to further investigate creating resources for Finnish and to continue using the SD scheme in these investigations. From this work we had gathered valuable data and experience, which were now put to use in developing what would eventually become the Turku Dependency Treebank.

2.3 The Turku Dependency Treebank

The second Finnish language corpus presented in this thesis, and likely its single most important contribution, is the *Turku Dependency Treebank* or *TDT*. This treebank consists of 204,399 tokens (15,126 sentences) of general Finnish. Of this amount, 10% on the level of documents are kept as a secret *test set*, which is only to be used for parser evaluation purposes and possible shared tasks in the future.

In the following, we will describe the most important aspects of the treebank: the text selection, the syntactic annotation and morphological information included, and the annotation protocol. We will also briefly discuss the development of the annotation scheme over time and its relationship with the English SD scheme. Finally, we discuss the baseline parser of general Finnish that was induced as part of the work of Paper VI.

2.3.1 The text of the treebank

The text of TDT has been selected with statistical parsing of general Finnish in mind. This leads to two important goals that manifest themselves as selection criteria for the text of the underlying corpus. First, to be maximally useful as a resource for parsing and other natural language processing, the treebank must be freely available, preferably not only for research, but for commercial purposes as well. This goal means that the text selected should be either originally published under a free licence, or alternatively it should be possible, with reasonable effort, to negotiate such a licence. Second, in order to sufficiently well represent general Finnish, the treebank must offer

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

Table 2.1: Sections of TDT and their sizes in terms of document, sentence and token counts. No document count is given for the grammar examples section, as this section consists of individual sentences with no further structure. *[Table from Paper VI.]*

linguistic variety. This means that texts from different text sources, on different topics and by different authors should be selected.

These two overall criteria have led to the selection of text from ten different text sources or genres: the Finnish Wikipedia and Wikinews, student magazines, blogs, a university magazine, a Finnish financial newspaper, speeches from the European parliament (from the Finnish section of the Europarl corpus (Koehn, 2005)), EU-legislation from the JRC-Acquis corpus (Steinberger et al., 2006), grammar examples from a Finnish reference grammar (Hakulinen et al., 2004), and amateur fiction. Each of these text sources forms a separate *section* in the treebank. The different sections and their sizes in different units are listed in Table 2.1.

The Wikipedia, Wikinews, university news, Europarl, JRC-Acquis, grammar examples and financial news sections contain text originally published under a free licence. The grammar examples were under a free licence via the FinnTreeBank project (Voutilainen and Lindén, 2011), and the financial news section consists of articles from *Taloussanomat*, which publishes the large majority of its articles under an open licence. We received the permission to re-license these particular articles to fit the licence of the treebank. In the remaining sections — blog entries, student magazines and fiction — each author was individually contacted in order to gain the permission to republish the work.

The variety of the treebank has been ensured in two ways: first, by selecting text from ten different text sources, and second, by limiting the amount of text selected by a single author or on a single topic. From each single text, such as a Wikipedia article or a blog entry, we have selected the first 75 sentences for annotation (or the whole text, if it is shorter than

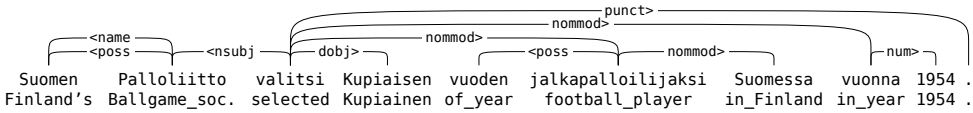

Figure 2.4: The SD scheme on general Finnish. The example can be translated as *The Finnish Ballgame Society selected Kupiainen as the Finnish football player of the year in 1954.*

75 sentences). This is in order to prevent the topics of long articles from being overrepresented in the treebank. In addition, in those genres where we have consciously selected multiple texts by a single author, we have limited the amount of text per author. In blog entries as well as amateur fiction, we have selected approximately 200 sentences by each blogger or fiction writer at most, and in student magazine articles, where typically the whole magazine is produced by a small number of active contributors, at most two articles per author.

2.3.2 The SD scheme on general Finnish

The work of Paper I showed that the SD scheme was suitable for ICU Finnish with certain minor modifications, which result was promising in the sense that we were encouraged to apply it to general Finnish as well. Thus SD was also used as the annotation scheme of the general language treebank. The syntax annotation of TDT contains two separate layers: the *base* layer where the analyses are trees and the dependencies are mostly syntactic in nature, and the *conjunct propagation and additional dependencies* layer, which adds dependencies on top of the base layer. The base layer of annotation is discussed below, and the second layer of annotation in Section 2.3.3.

As mentioned in Section 2.2.2, several of the modifications used in the clinical treebank could be directly exploited for general Finnish as well. These involve the treatment of adpositional phrases, passives and omitted head words. In contrast, the dependency type *sdep* used in the clinical treebank has not been used in TDT, as combining sentences with merely a comma or with no surface-marking whatsoever is considerably rarer in general Finnish. Also, in TDT punctuation was attached to the tree structure, unlike in the work of Paper I, where it was omitted from the analysis. (For the work of Paper II, the clinical treebank was augmented with punctuation annotation.) Figure 2.4 illustrates the SD scheme on general Finnish.

The *null tokens* of TDT, inherited from the annotation scheme of the clinical treebank, have a somewhat broader use in general Finnish. Whereas in the clinical treebank, all null tokens were verbs, in TDT they can also stand for omitted head words of other parts-of-speech, although verbs are still the most common case (651 out of 706 null tokens). In TDT, the null

Figure 2.5: Genitive subjects (left) and objects (right) in TDT. A Finnish derived noun or a noun with a verb counterpart can take a subject or an object in the genitive case. The examples can be translated as *The falling of the vase made him sad* and *The building of the house was a big project.*

tokens are used in two situations: in *fragments* and *gapping*.

Fragments, clauses with an omitted main predicate, such as *Presidentti Kiinaan (The president to China)*, are common in for instance newspaper titles. In these clauses, the null token is used to represent the missing main predicate, in the same way as in the clinical corpus. In cases of *gapping*, a head word has been omitted to avoid repetition, as in for instance *Äiti luki sanomalehteä ja isä kirjaa (Mother read a newspaper and father a book)*. In these cases, a null token is used to represent the elided token, and the elision is also marked with an additional dependency in the second annotation layer (see Section 2.3.3). Gapping is also present and marked in the clinical treebank, although relatively rare. The the second-layer dependencies signaling gapping were migrated from the early phases of TDT into the work of Paper II.

Note that gapping is only one of several different types of elliptical expressions, the elision of a head word. Other types of ellipsis are not marked using null tokens, as the elided token is not necessary to construct an analysis; however, some forms of ellipsis are covered by the *conjunct propagation* described in Section 2.3.3.

In addition to the modifications already introduced in the clinical treebank, certain small modifications were introduced in TDT as well. *Genitive subjects and objects* (Figure 2.5) of a noun were given their own dependency types, as well as *infinite clausal complements* (Figure 2.6) and *multi-word named entities* (Figure 2.7). *Vocatives* and *interjections* (Figure 2.8) do not typically occur in ICU Finnish, but for general Finnish they, too, required their own dependency types. In general, the modifications were small-scale, as they were in the clinical treebank. For further details on the annotation scheme of TDT, see the annotation manual by Haverinen (2012).

2.3.3 Conjunct propagation and additional dependencies

The base layer of TDT annotation described in the previous subsection requires the analyses to be trees, with the exception of the dependency type *name*, which is allowed to break the tree structure in the case of named entities with an internal structure (see Figure 2.7). This restriction, however,

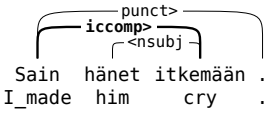

Figure 2.6: Infinite clausal complements are clausal complements with an infinite main verb, separately marked in the treebank. The example can be translated as *I made him cry*.

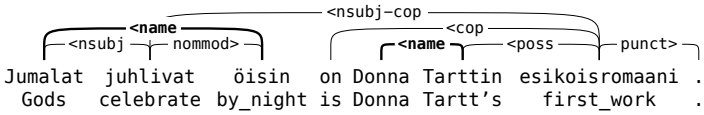

Figure 2.7: Multi-word named entities, illustrated by the example *Gods celebrate by night is Donna Tartt's first work*, where *Gods celebrate by night* is the Finnish title for *A Secret History*. The name of the book has an obvious internal syntactic structure, which is marked in the analysis, and the name of the author is merely a name. *[Figure from Paper VI.]*

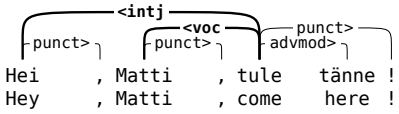

Figure 2.8: Vocatives and interjections were previously unaccounted for in the SD scheme but required an analysis in TDT. Both phenomena are exemplified by the sentence *Hey, Matti, come here!*.

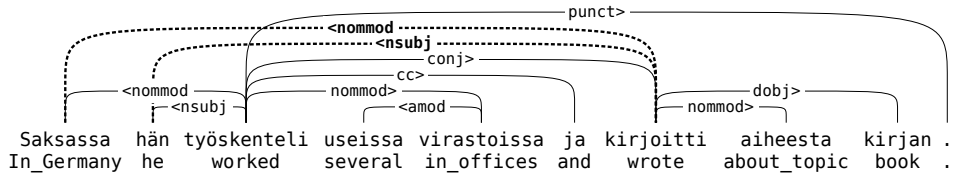

Figure 2.9: Conjunct propagation, exemplified by *In Germany, he worked in several offices and wrote a book about the topic.* The two coordinated verbs share the nominal modifier *Saksassa (in Germany)* and the subject *hän (he)* as shown by the dashed dependencies.

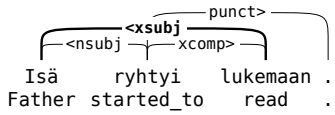

Figure 2.10: External subjects. The example can be translated as *Father started to read*, where *Father* is the subject of both verbs.

leaves some arguably important phenomena without a full analysis. Therefore TDT contains a second annotation layer, termed *conjunct propagation and additional dependencies*, which addresses this issue to some extent. As this layer adds dependencies on top of the tree structures, the resulting analyses are graphs rather than trees. The second layer annotation is part of the work presented in Paper VI, and a method for automatically predicting these dependencies has been presented by Nyblom et al. (2013). The utility of the second annotation layer in the construction of a Finnish PropBank is discussed in Paper VII.

The second annotation layer addresses four separate phenomena: *conjunct propagation*, *external subjects*, *syntactic functions of relativizers* and *gapping*. In the following, each of them is briefly described in turn.

Conjunct propagation The most important phenomenon covered in the second annotation layer is the *propagation of conjunct dependencies*, as it was termed by de Marneffe and Manning (2008b). In the SD scheme, coordination structures are analyzed so that the first coordinated element is the head word of the whole coordination, and the other coordinated elements as well as the conjunction (if present) depend on it. This means that it is not possible to distinguish between elements modifying only the first coordinated element and those modifying some or all other elements as well, and likewise for elements governing the head of the coordination. The coordination propagation makes it explicit which elements of the coordination modify or are modified by another element. See Figure 2.9 for an illustration.

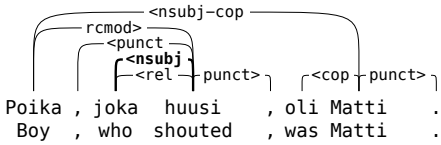

Figure 2.11: Syntactic functions of relativizers. In the example, *The boy who shouted was Matti*, the relativizer (*joka, who*) also acts as the subject of *huusi (shouted)*.

External subjects Open clausal complements are clausal complements that share the subject of their main verb, a phenomenon also known as *subject control*. The subject of the complement cannot be explicitly marked on the base layer of annotation due to the treeness restriction, and hence it is marked in the second layer using the dependency types *xsubj* for external subjects and *xsubj-cop* for external copular subjects (see Figure 2.10). External subjects interact with conjunct propagation both ways: external subjects may propagate in a coordination, and also propagated subjects may produce new external subjects.

Syntactic functions of relativizers The phrase containing the relative word (such as *which* or *who*) in a relative clause is marked simply as a *relativizer* in the first layer of annotation in TDT. However, the relativizer also always has a secondary syntactic function; for instance, it can act as the subject of the relative clause. Like external subjects, also relativizers may propagate in coordinations, and, if the secondary syntactic function of a relativizer is a subject, this subject may produce an external subject. Syntactic functions of relativizers are illustrated in Figure 2.11.

Gapping As mentioned in Section 2.3.2, only one type of ellipsis is explicitly marked in TDT: the elision of a head word, or *gapping*. In addition to marking the elided word with a null token, the word left unrepeated is the head of a dependency of the type *ellipsis* in the second annotation layer (Figure 2.12).

2.3.4 Scheme development and relation to the English scheme

An issue worth some consideration in the Finnish-specific SD scheme is its development over time. As discussed above, the scheme used for the clinical treebank was an early incarnation of essentially the same scheme that was later used for TDT. In addition, the scheme has evolved further during the annotation of TDT, and the current scheme of the treebank is that discussed in Paper VI and detailed in the annotation manual of Haverinen (2012).

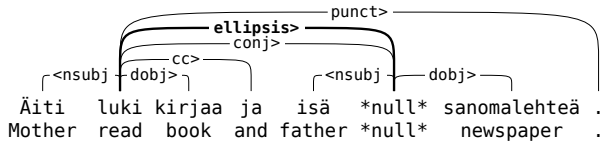

Figure 2.12: Gapping, exemplified by *Mother read a book and father a newspaper*. The missing head word is represented by a null token in order to be able to construct a dependency analysis, and in addition, the *ellipsis* dependency indicates that it is the verb *luki (read)* that has not been repeated.

For the most part, the changes made to the scheme were additions for phenomena that were only encountered once a wide enough selection of text sources were included. For instance, the early scheme discussions in Paper III do not include vocatives and interjections, because these phenomena were only encountered when annotating text from sources such as blogs and student magazines. However, also other changes occurred: for instance, in Paper III the *ellipsis* dependency is described as part of the base annotation layer (the only layer in existence at that point), but in the current release it has been moved to the *conjunct propagation and additional dependencies* layer, where it logically belongs.

This development over time shows that it is rather difficult, if not impossible, to set an annotation scheme in stone in the beginning of a project, but rather the scheme will evolve as necessary, when new examples are faced. In fact, if TDT were to be further expanded or adapted for a specific purpose — a perfectly possible scenario seeing for instance the recent extension of the treebank collection of McDonald et al. (2013) with TDT — it is possible that the scheme would again undergo slight changes.

From this perspective, it is also worth noting that not only the Finnish-specific SD scheme but also the English scheme version has changed over time. Shortly after the release of the current TDT version alongside with the publication of Paper VI, de Marneffe et al. (2013) have published an updated version of the English SD scheme, as part of a large SD-based annotation project of English (Silveira et al., 2014). Similarly to the development of the Finnish-specific scheme, the original SD scheme has evolved as the result of facing new genres with new phenomena, and the paper of de Marneffe et al. discusses the changes of the original scheme.

Interestingly, several of the changes proposed by de Marneffe et al. are exactly on par with the modifications we have made to the scheme for TDT, even though the scheme versions were developed fully independently of each other. The treatment of the previously unaccounted for phenomena of vocatives and interjections is (disregarding dependency type names) identical in the two schemes, and also the modifications of existing dependency types

included several changes also introduced in TDT. For instance, the discontinued use of purpose clause modifiers (*purpcl*), apposition-like abbreviations (*abbrev*) and attributives (*attr*) was perfectly in line with the Finnish-specific scheme.

The evolution of the SD scheme also has another, even more recent development. In the late spring 2014, there has been a collaborative effort for a new, *universal* version of SD (de Marneffe et al., 2014). This effort has been influenced by the emergence of several SD-based treebanks and the desire to further emphasize the design principle of language-independence. The universal SD uses ideas and relations used in existing SD-resources; for instance, the treatment of nominal modifiers and prepositional phrases is a generalization of the solution used in TDT. In general, the universal scheme version is (like the updated English SD version discussed above) quite compatible with TDT, and in fact, the paper of de Marneffe et al. (2014) outlines a conversion from the Finnish scheme to the universal.

2.3.5 Annotation protocol and quality

In this Section, we first discuss the overall protocol of annotation in TDT, and then move on to the annotation quality as measured separately for the two annotation layers. Finally, we discuss experiments on using a parser as an aid in the annotation process and the effect it has on annotation speed and accuracy.

The overall protocol

The entirety of TDT has been annotated using *full double annotation*. This means that each sentence has been independently annotated by two different annotators, and the resulting analyses have subsequently been merged together to form a single analysis for the sentence.

The annotation process consisted of two phases: first, the *base* layer annotation was created on top of text with undisambiguated morphological analyses, and second, the *conjunct propagation and additional dependencies* layer was annotated using the base layer annotation as a starting point. A total of seven different annotators took part in the treebank annotation: five in the first layer annotation and six in the second layer annotation.

The workflow in both of the annotation layers was similar, and consisted of the following three phases:

Individual annotations In the first phase of the work, each document of the treebank was assigned to two different annotators, who annotated the same document independently. This resulted in two separate analyses for each sentence, termed the *individual annotations*.

Merged annotation After a document had received both of its *individual annotations*, these analyses were automatically merged into one, where both alternatives are present and visually marked whenever there is a difference. Next, the annotation team held a meeting, and the best single analysis for each sentence was decided upon. The result of these meetings is the *merged annotation*.

Final annotation After the annotations had been merged, a round of consistency checks and corrections was performed on the *merged annotation*. This was to make sure that also old annotations conform to the newest annotation decisions, and where necessary, to correct errors in sentence splitting and tokenization.

The annotation protocols differ slightly between the two layers, with respect to the automatic aids used. In the first layer, an automatic parser was used to pre-annotate the text for one of the two annotators once an initial portion of the sentences had been annotated from scratch and a preliminary parser could be induced — the effects of this part of the protocol are discussed below. In the second layer, in turn, a pre-processing system was used to suggest possible dependencies to add, and both annotators used the output of the pre-processing as their starting point. Figure 2.13 illustrates the workflow of each of the TDT layers.

Quality of annotation

In order to evaluate and monitor the performance of the annotators and thus the quality of the annotation, we measured *annotator accuracy (AA)* of the *individual annotations* against the *merged annotation*. The *final annotation* was not used as a gold standard, seeing that we did not want to penalize an annotator for a decision that was correct at annotation time but that later became outdated, and also because the possible sentence splitting and tokenization corrections cause the *individual* and *final* annotations to no longer be directly comparable.

In the *base* layer of annotation, AA was measured using labeled attachment score. Preliminary results on AA were reported in Paper IV with the four annotators involved in the work so far.[7] The final results are reported in Paper VI, and repeated here in Table 2.2.

As can be seen from the Table, the overall AA across the annotators and text sources is 91.3%, indicating high annotation quality overall. The differences between genres seem to be smaller than the differences between annotators, and in addition, the figures may be affected by a learning effect. The table also shows the percentages of all tokens annotated by each

[7] At this stage, the measure was termed *inter-annotator agreement*, which was later substituted for the more precise term *annotator accuracy*.

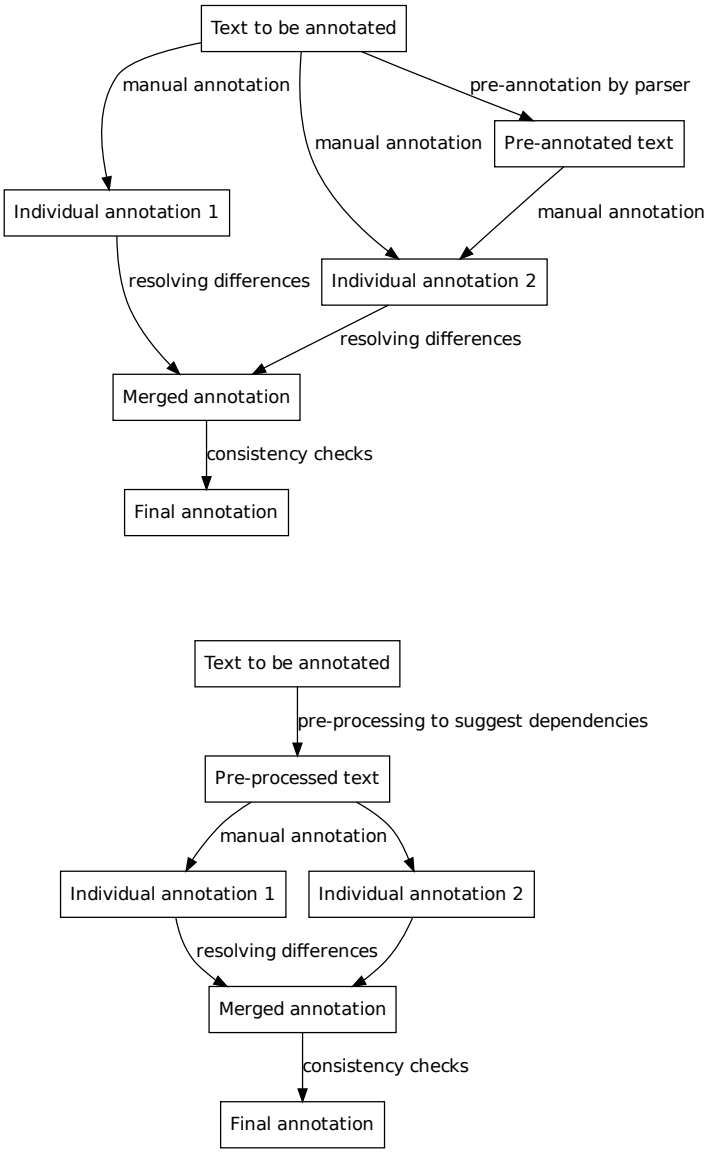

Figure 2.13: The annotation workflow of the *base* layer (top) and the *Conjunct propagation and additional dependencies* layer (bottom) of TDT. In the first layer, an initial portion of the text was annotated without any technical aids and later one of the annotators corrected the output of a preliminary parser, whereas in the second layer both annotators received a pre-processed text to annotate.

	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

Table 2.2: AA per annotator and per section. 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. *[Table from Paper VI.]*

annotator. Note that the maximum that one annotator could theoretically annotate is 50%, as each sentence must be annotated by two different annotators.

In the second annotation layer, we have measured AA using F_1-score, defined as $F_1=\frac{2PR}{P+R}$, where P stands for *precision* and R stands for *recall*. Precision is the percentage of dependencies present in the annotator output that are also present in the gold standard, and recall is the percentage of dependencies in the gold standard that are also present in the annotator output. The usage of the F_1-score here is due to the fact that in this layer, a large number of tokens are not considered in the annotation at all, which means that all these tokens would be counted as correct in LAS, and this in turn would result in artificially high figures. In addition, as the first layer analyses are required to be trees (with the exception of the dependency type *name*), the number of dependencies per sentence is fixed, and thus an annotator omitting a correct dependency will have to also add an erroneous dependency in order for the erroneous structure to still be a tree, which is not true for the second layer. LAS is intended to be used for tree structures, where precision, recall and F_1 would all be equal. The second layer AA results are also reported in Paper VI and repeated in Table 2.3.

Overall, the second annotation layer shows high accuracy, and even though not directly comparable with the first layer due to different measures, it would seem that the second layer annotation is the easier of the two tasks. Intuitively, this seems likely, seeing that instead of building a full tree of dependencies, the second layer annotation only requires deciding on the existence or non-existence of suggested dependencies and in some cases, their types.

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

Table 2.3: Evaluation of the second annotation layer given in precision (P), recall (R) and F_1-score. *[Table from Paper VI.]*

Pre-annotation and its effect on annotation accuracy and speed

As described in detail in Paper IV, part of the annotation has been done using a parser as an aid, in what we call a *pre-annotation* setting. This means that using the MaltParser system (Nivre et al., 2007), a preliminary parser was trained on the completed part of the treebank (at the time of writing Paper IV, a little under 50,000 tokens), and new annotation was done by first producing parser output for a document and then having an annotator correct this output. This approach was motivated by previous work by for instance Rehbein et al. (2009) and Fort and Sagot (2010), where the authors had found a similar protocol to have a positive effect on annotator speed or accuracy in different annotation tasks.

For each document annotated using the pre-annotation protocol, one annotator was given the parser output to correct and the other annotator was to annotate the same document from scratch. The dependencies produced by the parser were visually marked, so as to easily distinguish parts of the sentence already inspected by the annotator from those yet awaiting inspection.

In Paper IV, we explored any possible effect this protocol may have on the annotation speed and accuracy. This was done by comparing the accuracy (in LAS) and speed (in seconds/token) of an annotator between documents annotated from scratch and those annotated using the pre-annotation protocol, and then measuring statistical significance.

We found that whether pre-annotation is beneficial or harmful seems to depend on the annotator. Our least experienced annotator, Annotator 4, showed a rather tremendous increase in LAS when using the pre-annotation protocol; in fact, when annotating without the aid of a parser, the performance of this annotator was below that of our baseline parser, but given the output this parser, the annotator was able to clearly improve it as measured in LAS. On the other hand, our most experienced annotator, Annotator 1,

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	

Table 2.4: OMorFi output. The correct readings are marked by emphasis. The example sentence can be translated as *He doesn't live in a small village. [Table from Paper VI.]*

showed a small benefit in annotation speed but also suffered a minor decrease in LAS when using pre-annotation. For the other two annotators, there was no statistically significant effect on either speed or accuracy.

Based on these findings — the pre-annotation could be helpful to an inexperienced annotator and was not clearly harmful to others — the pre-annotation was continued throughout the construction of the first layer of TDT. Of the current treebank, over 10,000 sentences (nearly 150,000 tokens) have been annotated using the pre-annotation protocol.

2.3.6 Morphological analyses

The morphological analyses of the current version of TDT are based on an existing open source analyzer of Finnish, OMorFi (Pirinen, 2008; Lindén et al., 2009), which is part of the Open Source Morphologies (OMor) project by the University of Helsinki. As this tool produces undisambiguated morphological readings of high quality, and as the cost of manual annotation is high, we have used an automated method to produce the morphological analyses of TDT by disambiguating the readings provided by OMorFi.

The output of OMorFi, in the output mode used in TDT, is illustrated in Table 2.4. In order to be able to utilize this output, two issues require solving. First, approximately 48.6% of all tokens in TDT are ambiguous by OMorFi, that is, they have more than one reading. These ambiguities need to be resolved, so that the final analysis consists of one reading for each token. Second, 5.2% of TDT tokens are not recognized by OMorFi, and therefore their analyses need to be either manually annotated or otherwise inferred.

The treatment of these issues can be divided into two phases, as described in Paper VI. In the first phase, the OMorFi output is post-processed, so as to discard readings that are, for practical purposes, equivalent. OMorFi fre-

quently produces such readings for derivations and compounds. For instance for the wordform *tekeminen* (*doing*, noun) OMorFi produces two readings: one indicating that *tekeminen* is a *-minen* derivation of the verb *tehdä* (*to do*) and as such a noun, and the other directly analyzing it as a noun. Judging which one of these readings is correct is highly difficult, as quite plausibly, it can be claimed that they both are. In these cases, the post-processing selects one reading and discards all of the readings considered equivalent with it.

In addition to the post-processing, the first phase contains the treatment of unknown tokens. Unknown compound tokens constructed using a dash are given the analysis of their latter component if this component alone receives one, as in Finnish, the last part of a compound determines its category. In addition, regular expressions are used to find tokens that are symbols rather than words. Finally, multi-word named entities with no internal structure inherit a coarse-grained analysis from their head word, and all remaining unknown tokens (including the head words of multi-word named entities) are manually annotated.

The first phase reduces the ambiguity present in the treebank, as well as gives an analysis to all unknown tokens. As a result, after the first phase of morphology treatment, 62.9% of all tokens in TDT are unambiguous, and the rest are ambiguous. As the second phase of the morphology treatment, we use a machine learning method to resolve the remaining ambiguity.

This method relies on two important insights. First, morphological ambiguity in unsystematic: for instance, although the wordform *koirasta* is ambiguous between the singular partitive of *koiras (male)* and the singular elative of *koira (dog)* most other elative and partitive wordforms are not ambiguous in this way. Second, the existing syntactic annotation of the treebank provides cues for disambiguating the morphological readings. For instance, it is quite unlikely for a token acting as a nominal subject to be a verb.[8]

Using these two insights, the disambiguation of the morphological readings is cast as a machine learning problem: the alternative readings of each ambiguous token are to be ranked, and the highest-ranking reading is to be selected for the token. The method uses unambiguous and partially ambiguous tokens as training examples, and its features are generated from the readings of the tokens, the syntactic trees, and agreement. After the machine learning method has been applied, a small set of rules is used to address certain consistently ambiguous cases.

The morphology disambiguation method is evaluated on a manually annotated set of 1,000 tokens randomly selected from the treebank test set.

[8] *quite unlikely* rather than *impossible*, as a verb participle can act as the head of a noun phrase in TDT in rare cases (see Section 5.8 on *Noun phrases without nouns* in the manual of Haverinen (2012))

	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

Table 2.5: Morphology assignment evaluation in terms of accuracy. 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 Paper VI, 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. *[Table from Paper VI.]*

These tokens were sampled from all other tokens except numbers and punctuation. The results of the evaluation are shown in Table 2.5. The evaluation is given in terms of *accuracy*, defined as the proportion of correctly assigned readings, and on three different granularities: first, only considering the main POS tags, second the POS and all other morphological tags, and finally, all tags and also the lemma. Results are listed separately for unambiguous and ambiguous tokens, as well as for tokens unknown to OMorFi and the null tokens added during annotation, which also receive an analysis in TDT. Finally, an overall evaluation is given (*All*), as well as an estimate of the overall accuracy including numbers and punctuation. This last evaluation is performed by calculating an average accuracy weighted by the proportion of numbers and punctuation in the corpus and assuming a perfect accuracy for these tokens.

2.3.7 Parsing general Finnish

In addition to TDT, Paper VI presents as part of its main contribution the first freely available statistical parsing pipeline of Finnish. The pipeline consists of the standard components of sentence splitting, tokenization, POS and morphological tagging, and dependency parsing.

For sentence splitting and tokenization, the pipeline utilizes modules from the Apache OpenNLP tools,[9] and for POS and morphological tagging, a combination of OMorFi and the statistical tagger HunPOS (Halácsy et al., 2007) is used. For any token recognized by OMorFi, HunPOS is given the

[9] `http://opennlp.apache.org`

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 2.6: Parsing results for Finnish on TDT. *[Table from Paper VI.]*

task of selecting one of the readings given, and for any token unrecognized by OMorFi, HunPOS is used to guess the most likely tags based on the suffixes of the word.

After the sentence splitting, tokenization and morphological analysis, the text is dependency parsed using the state-of-the-art parser of Bohnet (2010), trained in a standard manner. The results achieved by the parser are reported in Paper VI and repeated in Table 2.6. The overall performance of the parser is 81.01% LAS on the test set of the treebank. As illustrated by Figure 2.14, which gathers together parsing results from several recent comparison studies, this is in the expected range for a treebank of the size of TDT. In addition to the result of Paper VI, this figure now includes the result of Bohnet et al. (2013), who have been able to improve the state of the art of parsing Finnish with a LAS of 83.1% slightly after the full release of TDT.

TDT and the parsing pipeline have already been used in other NLP projects. As described in the paper by Ginter et al. (2013), an almost final subset of sentences from TDT has been used to produce a parsebank from the Finnish sections of the Europarl (Koehn, 2005) and JRC-Acquis (Steinberger et al., 2006) corpora. This work was done as contract research for the FinCLARIN consortium, and the resulting parsebank is distributed by the University of Helsinki under the name FinnTreeBank 3. TDT has also been used in a project aiming to produce another, even larger parsebank consisting of Internet texts. This project is still ongoing, but the first results are already becoming available (Kanerva et al., 2014; Kanerva and Ginter, 2014). In addition, TDT has been used in a machine translation project in collaboration with Convertus AB.[10] The purpose of this project was to build a machine translation system from Finnish to English, with a focus on the domain of education. Also in the field of parsing itself, TDT has already sparked new interest in the Finnish language, as demonstrated by the work of Bohnet et al. (2013), and in scheme design, TDT has, for its part, helped inspire the new, universal version of the SD scheme (de Marneffe et al., 2014).

[10] `http://www.convertus.se/`

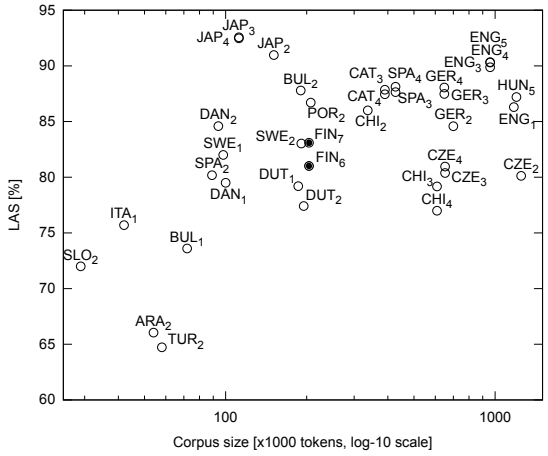

Figure 2.14: Parsing results for various languages from a number of studies, as related to corpus size. The languages are given as short labels, where CAT=Catalan, CHI=Chinese, CZE=Czech, ENG=English, GER=German, JAP=Japanese SPA=Spanish, ARA=Arabic, BUL=Bulgarian, DAN=Danish, DUT=Dutch, POR=Portuguese, 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), 6=Paper VI, 7=Bohnet et al. (2013). The Finnish parsing results are shown as black dots. *[Figure adapted from Paper VI.]*

2.4 Annotation error analysis

In addition to the final treebank annotations, the Turku Dependency Treebank also provides another type of data, data that can be utilized to study the annotation process itself. TDT is somewhat exceptional in being constructed using full double annotation; many treebanks, for instance the well-known Penn Treebank (Marcus et al., 1993), are single-annotated after an initial training phase. As was described in Section 2.3.5, the double annotation of a document produces three different kinds of data: the *individual, merged and final annotations*. Paper V uses these data to study the annotation process and the different errors produced by the annotators. This paper also uses timing data gathered during annotation to further evaluate the difficulty of annotation.

The error analysis of Paper V examined the portion of the treebank completed in the spring of 2011, that is, a subset of 100,073 tokens in 7,076 sentences, out of which 10% on the level of documents were only used for parameter optimization in the experiments involving parsing. In this phase

of the annotation process, there were in total five annotators contributing to the treebank, with different backgrounds and previous knowledge. Also, at this time, the study could only concern the base annotation layer, as the *conjunct propagation and additional dependencies* layer was only added to the treebank after the whole treebank had first been given the base layer annotation.

2.4.1 Typical errors and quality of the *final annotation*

The first part of the study in Paper V discusses the most difficult dependency types to annotate, defined as dependency types for which the F_1-score was the lowest. As also in Paper VI, the evaluation was performed against the *merged* annotation, not the *final* annotation, and for the same reasons. The most difficult dependency types for human annotators and two parsers induced using the MaltParser system (Nivre et al., 2007) and the MateTools parser (Bohnet, 2010) were reported in Paper V, and are repeated in Table 2.7.

Many of the dependency types listed as difficult for humans in Table 2.7 describe different complements: *iccomp* and *ccomp* are for clausal complements, and their distinction relies on the morphological form of the head. Similarly, *acomp* is for adjectival complements of a verb. Distinguishing complements from modifiers, and in the case of clausal complements, distinguishing between their different types, may be causing difficulties. Another type possibly suffering from the difficulty of the complement and modifier distinction is *advcl*, intended for *adverbial clause modifiers*. In addition to these distinctions, all of the above types may occasionally also suffer from the difficulty of attaching the dependent to the correct token; it may in some cases be difficult to say what exactly is being modified.

Appositions (*appos*) and genitive objects (*gobj*) require making highly semantic distinctions, while on the other hand, genitive objects may also be subject to simply being overlooked in favor of the more general type for genitive modifiers, *poss*. A similar note applies to *auxpass*, which is intended for passive auxiliaries. This type can easily be forgotten and replaced by the more general auxiliary type *aux*. Also the phenomena of parataxis and comparative structures (the type *compar*) are among the more difficult ones to human annotators, possibly due to parataxis resembling coordinations and comparative structures often being elliptical. Also the most general dependency type of SD, *dep*, which in TDT is mostly used in certain fixed multi-word expressions, proved rather difficult for the annotators.

The two parsers had some of the same difficulties as human annotators, with for instance different complements, but some of their errors differed from those of humans. For instance, coordinations (the dependency type *conj*) were difficult for MaltParser, and the type *nn*, used for compounds

Human

type	P	R	F
iccomp	68.8	70.9	69.8
parataxis	69.9	71.6	70.7
acomp	74.1	70.5	72.2
compar	77.0	71.6	74.2
dep	85.8	69.4	76.7
advcl	79.2	79.1	79.2
auxpass	84.9	75.7	80.0
ccomp	82.2	79.4	80.8
appos	81.7	80.2	81.0
gobj	88.6	77.4	82.6
overall	89.9	89.1	89.5

MaltParser				**MateTools**			
type	**P**	**R**	**F**	**type**	**P**	**R**	**F**
parataxis	24.2	8.3	12.4	gobj	65.9	26.5	37.8
dep	13.3	34.0	19.1	parataxis	43.0	38.0	40.4
advcl	34.7	39.4	36.9	appos	41.4	40.7	41.0
appos	40.1	38.2	39.1	compar	49.7	44.1	46.7
compar	61.4	35.0	44.6	acomp	52.2	45.5	48.6
acomp	53.2	43.5	47.9	iccomp	51.4	48.8	50.1
rcmod	49.9	48.5	49.2	dep	60.8	43.3	50.6
ccomp	57.0	49.4	52.9	advcl	54.4	49.6	51.9
iccomp	63.7	48.0	54.8	nn	62.1	55.8	58.8
conj	61.0	63.0	62.0	ccomp	63.9	62.9	63.4
overall	71.5	71.2	71.3	**overall**	74.4	74.7	74.6

Table 2.7: The ten hardest dependency types for human annotators and the two parsers. The standard F_1-score was calculated for each dependency type separately, considering only those types that occur in the gold standard at least 150 times. This table presents the ten dependency types with the lowest F_1-scores. For each type is given its precision, recall and F_1-score. *[Table adapted from Paper V.]*

and appellation modifiers, for MateTools. Interestingly, out of the ten most difficult dependency types, eight were shared between the two parsers, even with the different parsing paradigms underlying them.

As the second part of the study presented in Paper V, we have examined cases where annotators have confused two dependency types with each other, that is, where the governor and dependent of a dependency were correct, but the dependency type was incorrect. In total 32.4% of all erroneous dependencies were type confusions, and the most common type confusions of the different human annotators could be divided into categories of errors. These most common confusions are given in Table 2.8.

One class of errors were cases where the annotator confused morphologically and semantically similar phenomena. One common example of this error type is the confusion of *direct objects* (*dobj*) and *nominal modifiers* (*nommod*); certain complements of the verb semantically resemble objects although due to case restrictions are not considered such, and in addition, so called *object-cased amount adverbials* even take the same cases as objects.

Annotator 1		Annotator 2		Annotator 3	
GS type	**annot. type**	**GS type**	**annot. type**	**GS type**	**annot. type**
advmod	nommod	dobj	nommod	advmod	nommod
dobj	nommod	gobj	poss	dobj	nommod
auxpass	aux	nsubj	dobj	nommod	dobj
gobj	poss	advmod	nommod	advmod	advcl
nommod	advmod	nommod	dobj	nommod	appos

Annotator 4		Annotator 5	
GS type	**annot. type**	**GS type**	**annot. type**
dobj	nommod	dobj	nommod
nommod	dobj	acomp	nommod
gobj	poss	partmod	advcl
nsubj	nommod	appos	conj
nsubj	dobj	nommod	dobj

MaltParser		MateTools	
GS type	**annot. type**	**GS type**	**annot. type**
gobj	poss	gobj	poss
nommod	dobj	nommod	dobj
partmod	amod	dobj	nsubj
name	poss	name	poss
dobj	nsubj	dobj	nommod

Table 2.8: The five most common dependency type confusions for each annotator and the two parsers. For each confusion is given the gold standard dependency type (GS type) and the type suggested by the annotator/parser (annot. type). *[Table adapted from Paper V.]*

Similarly, due to their semantic and morphological closeness, subjects and objects were confused by some annotators. The latter confusion may at first glance seem surprising, but due to the free word order of Finnish as well as the fact that subjects and objects take the same cases, being misled especially in fast-paced annotation is rather understandable. The similarity of Finnish subjects and objects can be illustrated by the ambiguity of the sentence *Pankkiautomaatit räjäyttivät todennäköisesti samat varkaat*,[11] a genuine newspaper title that has caused some public amusement. Due to an ambiguity on the part of subjects and objects, the sentence can be read as either *The same thieves likely exploded the ATMs*, as intended, or *The ATMs likely exploded the same thieves*, as would be suggested by the word order.

The second class of errors consisted of those based on a difficult morphological distinction that is needed also on the level of the syntactic annotation. Such distinctions in Finnish are for instance those between nouns and adverbs or adjectives and participles — especially the latter is also difficult in English (see for instance the Penn Treebank POS annotation manual (Santorini, 1990, p. 14)).

Finally, the third group consisted of typographical errors. Here there was no linguistic confusion, but rather those annotators who used the shortcut keys in the annotation software occasionally selected a dependency type whose shortcut key was adjacent to that of the correct type, or confused types whose keys were capital and non-capital versions of the same letter.

Both MaltParser and MateTools also produced confusion errors, includ-

[11] Yle uutiset 14.8.2009, title changed since.

ing both types of linguistic confusions present in human annotation, but naturally, they made no typographical errors. Again, the two parsers behaved similarly: they shared four out of the five most common dependency type confusions.

The study also compared human annotators and the baseline parser in order to find whether annotation errors co-occur with parsing errors with respect to their position. The results of Paper V show that there is a clear association between the two, much higher than would be expected by chance. This indicates that there are some structures that are in this sense "universally difficult".

Next, Paper V presented an evaluation of the *final* annotation of TDT. While AA as calculated between the *individual* annotation and the *merged* annotation gives an estimate of the quality of the *individual* annotations, it does not directly measure the quality of the annotation present in the treebank, that is, the *final* annotation. Our experiment on this matter in Paper V is as follows: One annotator independently re-annotated a random sample of 100 sentences from the treebank, such that this annotator had not previously annotated them. Subsequently these annotations were merged with the *final* annotations of the treebank using the usual procedure, and this new merged annotation then served as a gold standard against which the *final* annotation could be measured.

The resulting LAS for the *final* annotation was 98.1%, further confirming good annotation quality. Together with the then AA of 89.2% it indicated that approximately 82% of errors remaining in the *individual* annotations could be weeded out using double annotation. This experiment was later repeated in Paper VI, using the two best annotators to annotate a sample of 200 sentences each (the best annotator had already annotated a substantial portion of the treebank and thus could not be used to evaluate the majority of sentences) and the results were similar: with an overall AA of 91.3% and a LAS of 97.6% in the triple annotation experiment, approximately 72% of errors could be eliminated using double annotation.

2.4.2 Predicting annotation errors

The final part of the contribution of Paper V is a machine learning method intended to aid annotation projects in inspecting sentences for errors. In a compromise setting where only some sentences are double annotated or otherwise carefully inspected, if one selects sentences for inspection at random, it is expected that the amount of annotation errors found is proportional to the portion of the data inspected. For instance, if one inspects 10% of all sentences in a treebank, it is expected that 10% of all annotation errors are presented to the inspection. Therefore the goal is to provide a better method at selecting sentences to be inspected, in order to reduce the effort

required for visiting a certain proportion of the errors.

Paper V thus presents a method that, using as features morphological and syntactic properties of the tree produced by the annotator, as well as the identity of said annotator, provides a ranking of sentences based on their likelihood of containing annotation errors. When evaluated against the baseline of selecting sentences for inspection randomly, we find that the method achieves a clearly superior performance. For instance, by inspecting 25% of all sentences using this method, one is able to correct 50% of all annotation errors, whereas by selecting 25% of all sentences for inspection randomly, only 25% of errors would be found.

Chapter 3

PropBanks

This Chapter discusses the semantic aspects of the corpora presented, that is, the Proposition Banks created on top of the two treebanks described in the previous Chapter. Again, we begin by discussing the annotation scheme used. Then we turn to the clinical language corpus and the PropBank presented in Paper II, and afterwards, we will discuss the general language PropBank annotated on top of Turku Dependency Treebank. The annotation of the latter PropBank has only rather recently been completed, and the manuscript describing the complete work is under preparation. Paper VII is the first description of the Finnish PropBank, as a work in progress.

3.1 The PropBank annotation scheme

For the semantic annotation of the two corpora presented in this thesis, we use the PropBank annotation scheme, originally developed for English by Palmer et al. (2005). This choice is a rather popular one: PropBanks for different languages have been constructed after the appearance of the original work, including for instance PropBanks for Chinese (Xue and Palmer, 2009), Arabic (Zaghouani et al., 2010), Hindi (Palmer et al., 2009) and Brazilian Portuguese (Duran and Aluísio, 2011). Our choice is motivated by the practical approach of the scheme, where instead of attempting to decide on universal semantic roles, the problem is tackled one verb at a time. In addition, the PropBank scheme is intended for running-text annotation of corpora, whereas the other two well-known SRL resources mentioned in Section 1.3, FrameNet (Baker et al., 1998) and VerbNet (Dang et al., 1998), are rather intended to be lexical resources.

The PropBank scheme defines semantic roles separately for each verb, using numbered labels, such as *argument 0* (*Arg0*) and *argument 3* (*Arg3*). Each verb receives one or more *framesets*, which define the verb's arguments and which can be thought of as corresponding to coarse-grained senses of

break.01: break, cause not to be whole		**break.02**: gain entry	
Arg0	breaker	Arg0	enterer
Arg1	thing broken	Arg1	place/domain broken into
Arg2	instrument		
Arg3	pieces		

Figure 3.1: Two PropBank *framesets* for the English verb *to break*. These two framesets are to be used for different senses of *to break*; break.01 could be used for an example such as *John broke the piggy-bank into small pieces with a hammer* and break.02 for *John broke into the warehouse in the dead of the night*.

the verb. Figure 3.1 illustrates two framesets for an English verb. The first two arguments, Arg0 and Arg1, have special, predefined meanings: Arg0 is reserved for *agents*, *causers* and *experiencers*, while Arg1 is used for *patients* and *themes*. The arguments two to five, in turn, have meanings defined for each verb separately. However, the original PropBank strives to have consistent argument numberings within verb classes defined in the VerbNet project.

In addition to the numbered arguments, PropBank also defines a number of *adjunct-like argument* (*ArgM*) types, which, unlike the numbered arguments, can occur with any verb. The original English PropBank defines 11 different types for ArgMs, such as *cause* and *direction*. The distinction between numbered arguments and ArgMs is made based on frequency: if an argument candidate frequently occurs together with the verb sense under consideration, then it is granted numbered argument status, otherwise it is left as an adjunct-like argument.

As a PropBank consists of two parts, a verb lexicon and an annotated corpus, a two-phase workflow for constructing the resource is needed as well. First, for each verb the possible arguments and their definitions must be given, and second, the occurrences of the verb in the underlying text corpus must be annotated according to these definitions.

3.2 The clinical PropBank

The clinical PropBank is based on the treebank presented in Paper I. For the purposes of the work of Paper II, the text corpus and its syntax annotation from Paper I has been expanded: the full size of the treebank used in Paper II is 2,081 sentences with 15,335 tokens. These sentences comprise a total of eight complete patient reports from an intensive care unit. At the time of conducting the work of Paper II, we gained the permission to make an anonymized version of the text and all of its annotation publicly available.

As the clinical treebank does not contain gold-standard morphological

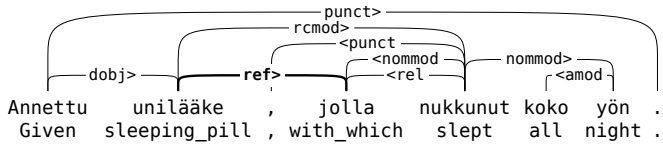

Figure 3.2: The annotation of *referents (ref)* in the clinical treebank. The example can be translated as *Given a sleeping pill, with which [the patient] has slept all night.*

information, the work of Paper II uses the FinCG analyzer to identify all verbs and verbal participles to be annotated. In total 2,816 tokens receive a verbal or participal reading. All verbs that have at least three occurrences were annotated, amounting to 2,382 tokens, which belong to 157 different verb lemmas with 192 framesets. The verbs of the corpus have a total of 4,763 arguments and ArgMs. As this project was relatively small-scale, the annotation of the resource was collaborative, meaning that the framesets were created and the occurrences were annotated by the annotation team together, and all disagreements were solved on the same occasion. This was to simplify the annotation as far as possible due to schedule restrictions, and therefore no annotator accuracy figures were reported for this resource.

3.2.1 Syntax annotation in the Extended SD scheme variant

As described above in Section 2.2.2, the annotation scheme of the clinical treebank of Paper I is a modified version of the SD scheme. For the work of Paper II not only the text of the corpus, but also the annotation described in Paper I has been extended. For the purposes of annotating the PropBank, we have added a second syntax annotation layer, similar to the one present in TDT (see Section 2.3.3). The annotation including the additional dependencies in the second layer is termed the *Extended* variant of the SD scheme in Paper II. The final release of the corpus also contains correctly attached punctuation, differing from the description given in Paper I.

The *Extended* annotation of the clinical treebank differs slightly from the *conjunct propagation and additional dependencies* annotation in TDT. The annotation in both resources contain the propagation of conjunct dependencies, external subjects, the syntactic functions of relativizers and gapping annotation (see Section 2.3.3 for more details). In addition, the clinical treebank contains annotation of *referents* in relative clauses, indicating the word which the relativizer refers to. Referents are illustrated in Figure 3.2.

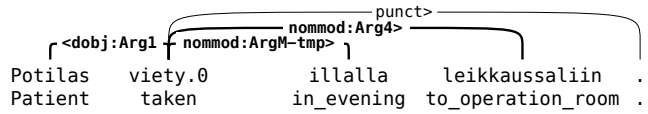

Figure 3.3: The clinical PropBank annotation. The verb *viety (taken)* has been assigned the frameset number zero, and arguments are associated with dependencies. The example can be translated as *Patient taken to the operation room in the evening.*

3.2.2 PropBanking on top of a dependency treebank

In contrast to the original PropBank (Palmer et al., 2005), where arguments are associated with nodes of the constituency-based Penn Treebank, both the clinical and general language Finnish PropBanks are built on top of a dependency treebank annotated in the SD scheme. This requires a different approach to associating the PropBank arguments to the underlying tree, and we use a similar solution in the case of both PropBanks.

In our approach, the arguments of the PropBank are associated with dependencies. This can be interpreted so that the dependent and its full subtree forms the argument. (Here *subtree* is defined by the first syntax annotation layer.) This approach is not entirely unproblematic; see Section 3.3.1 for discussion of its issues in the context of the Finnish PropBank. In the clinical treebank, we further restrict the arguments to direct dependents of the verb, in order to simplify the annotation process. The annotation of the clinical PropBank is illustrated in Figure 3.3.

In order to justify the restriction of arguments to direct dependents of the verb, we have annotated a sample of 100 sentences with all arguments regardless of their syntactic position. We find that in this sample, 91% of all arguments (be it numbered arguments or ArgMs) are direct dependents of the verb if both annotation layers are taken into account. Further, when considering numbered arguments only, 96% were direct dependents of the verb, and out of the ungoverned ArgMs, all were of the type *CAU (cause)* or *CSQ (consequence)*, which require strong inference. Based on these findings, our decision to restrict the annotation to direct dependents of the verb appears to be justified. Moreover, the fact that 6% of all arguments, on top of the 85% that are associated with a dependency on the first syntax annotation layer, are associated with an *extended* SD dependency demonstrates the utility of the additional syntax annotation.

3.2.3 Clinical Finnish and the PropBank scheme

Altogether, the PropBank annotation scheme appears to be suitable for clinical Finnish. The only overall change to the annotation scheme required

by the text was the addition of one ArgM-type, *CSQ (consequence)*. This addition was because not all kinds of causal relationships could be expressed using the existing type *CAU (cause)*, especially as only direct dependents of the verb were annotated. The *CSQ* type was also found to be useful for the annotation of the general Finnish PropBank and was preserved in the work of Paper VII.

One particular feature of the ICU language deserves a specific mention in this context. As discussed in Section 2.2.1, the nature of the language is quite fragmentary and telegraphic. This means that omissions of sentence elements, including the main verb, are common. Thus the most common verb of the corpus, with 428 occurrences, is in fact the *null token*[1] added during the syntax annotation.

As the *null tokens* make a substantial portion (about 18%) of all verbs in the corpus, an annotation strategy for them was required. Even though the null tokens in fact correspond to several different verbs (and in TDT, not all of them are even verbs), we have treated the null token as if it was a regular verb, and given it framesets accordingly.

In the framing and annotation of the null token, we found that 94% of all occurrences could be annotated according to only four framesets. These were the same framesets as were used for four other verbs: *olla (to be)*, *tulla (to come)*, *tehdä (to do)* and *laittaa (to put)*. The remaining occurrences were assigned to a so called *leftover frameset*, for which no arguments were defined.

3.3 The Finnish PropBank

We now turn to the work of Paper VII, which reports the first results from the construction of a PropBank for general Finnish based on TDT. The design principles of this PropBank are similar to those of the clinical PropBank: arguments are associated with dependencies from both syntax annotation layers, and the creation of framesets follows guidelines set by the original work on English. As mentioned above, the new ArgM type *CSQ* inserted in the clinical PropBank is used again in this work. Only one additional ArgM type is used in the Finnish PropBank: *PRT*, which is meant for the phrasal component of phrasal verbs, such as *up* in *look up*.

3.3.1 Dependency-based PropBanking and boundary issues

As with the clinical PropBank described in Section 3.2, we use both annotation layers of TDT to associate dependencies with, making use of the assumption that the full subtree of the dependent acts as the argument.

[1] termed *null verb* in Papers I and II

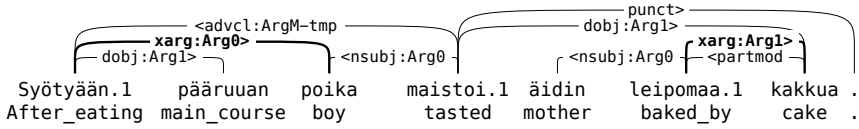

Figure 3.4: The general Finnish PropBank and external arguments. Arguments that are not direct dependents of the verb are marked using an *xarg* dependency inserted during PropBank annotation. The example can be translated as *After eating the main course, the boy tasted the cake baked by mother*.

Contrary to the clinical PropBank, however, we do annotate all arguments, regardless of whether or not they are direct dependents of the verb. Here we also annotate all verbs, regardless of how many occurrences they have. For arguments that are not dependents of the verb in either syntax annotation layer, a dependency of the type *xarg* (for *external argument*) is inserted during PropBank annotation, and the argument is associated with this dependency. This strategy is illustrated in Figure 3.4.

As mentioned above in the context of the clinical PropBank, the strategy of associating arguments with dependencies is not fully unproblematic. Choi and Palmer (2010) have presented work where they create a dependency-based PropBank by converting the Penn Treebank into a dependency format and subsequently retrieving semantic roles from the English PropBank, which has been annotated on top of the treebank. They find that although many dependency relations have high correlations with semantic roles (for instance, subjects with agents) and thus dependency structures are very suitable for representing predicate–argument structures, the boundaries of arguments as determined by dependency structure are not always correct.

Consider Figure 3.4 again, and the rightmost *xarg* dependency in particular. According to the assumption that the full subtree of the dependent word acts as the argument, the Arg1 argument of *leipomaa (baked by)* would in fact include the verb *leipomaa* itself, which is naturally incorrect; the correct argument would include only the word *kakkua (cake)*.

Choi and Palmer list a number of different syntactic cases where the semantic boundaries as retrieved directly from the dependency structure are incorrect. Fortunately for the work of Paper VII, several of these cases are unproblematic for the SD scheme. For instance, in the dependency scheme used in the work of Choi and Palmer, modal verbs act as the head word of their main verbs, which is inconvenient for a PropBank, where an auxiliary should become an *ArgM-MOD* for its main verb. This situation is unproblematic in the SD scheme, where an auxiliary always depends on the main verb. Similarly, in the scheme used by Choi and Palmer it is possible that a negation becomes the head of a main verb in certain coordination

structures, which causes the same problem. Again, this is not the case for the SD scheme, where negation depends on the main verb regardless of coordinations.

However, some situations, such as the one illustrated in Figure 3.4, are also problematic for the SD scheme. These issues are not resolved in the work of Paper VII, but we recognize that if one were to use the PropBank for an application that benefits from or even requires knowing exact argument spans, the remaining cases would need to be resolved. We envision that a rule-based solution would suffice, seeing that the boundary issues form certain regular cases. The same approach could naturally be used also to resolve these issues in the clinical PropBank.

3.3.2 Finnish-specific issues in frameset creation

The framesets of the Finnish PropBank are created using the same overall guidelines that were used in the creation of the original English PropBank: framesets correspond to coarse-grained senses of the verb, and where two potential framesets either have the same arguments (including argument descriptions) or the arguments of one are a subset of the arguments of the other, only one frameset should be created. See the framing manual (Babko-Malaya, 2005a) and the annotation manual (Babko-Malaya, 2005b) of the original PropBank for detailed guidelines.

In addition, the framesets of the English PropBank were used as models for Finnish ones where appropriate, and served as general guidance. In the work of Paper VII, this modeling was explicitly marked on the framesets, using the term *corresponding framesets*. Later the practice of explicit marking of corresponding framesets was discontinued due to time restrictions and the fact that the annotation software did not require annotators to fill in this particular field, which caused it to be easily forgotten. However, the similarities and differences between Finnish and English framesets remains an interesting point in the PropBanking work. Figure 3.5 illustrates identical framesets with a simple example using the Finnish verb *erota*, which can be translated as *to quit*.

erota.2: leave a job		**quit.01: leave a job**	
Arg0	Person quitting	Arg0	Person quitting
Arg1	Job or position	Arg1	Job or position

Figure 3.5: Finnish and English verbs with identical framesets. The Finnish verb *erota* can be translated as *to quit*, and the framesets of this verb sense define identical argument structures. *[Figure adapted from Paper VII.]*

As discussed in Paper VII, not all framesets are identical to any English frameset, even if a good translation exists. This may be partly due to differences between the Finnish and English languages, and partly simply due contextual differences between the texts of Penn Treebank and Turku Dependency Treebank.

One example of a particular feature of Finnish that causes systematic differences between Finnish and English framesets are *causative derivations*. In English, certain verbs, often movement verbs, are polysemous in a systematic way. One example is the verb *to move*, which can be used with an agent subject (*I move*), with a patient subject (*the chair moves*) or with an agent subject and a patient object (*I move the chair*). These verbs are often termed *variable behavior verbs* (see for instance the work of Levin and Hovav (1994) and Perlmutter (1978)).

Some verbs in Finnish behave similarly to the English variable behavior verbs, one example being *lentää (to fly)*, which can be used to describe a bird or a plane flying just as well as a pilot flying a plane. Most Finnish verbs, however, do not exhibit this behavior. For instance, the verb *liikkua*, which can be translated as *to move*, is intransitive, and thus can be used in contexts such as *minä liikun (I move)* or *tuoli liikkuu (the chair moves)*, but not in ones describing an agent moving a patient: **minä liikun tuolia (*I move(intrans.) the chair)* is ungrammatical. Instead, an agent moving a patient is expressed using a different verb, a so called *causative derivation* of the root verb *liikkua*: *minä liikutan tuolia*, literally, *I make the chair move*.

This difference between the languages has a consequence for the frameset creation. Verbs like *liikuttaa (to make something move)* can receive a frameset resembling the frameset for *to move*: both Arg0 and Arg1 are present. However, the frameset for *to move* is not quite identical to that of *liikuttaa*, as *to move* can have an agent or a patient as its subject, whereas *liikuttaa* must have an agent subject. As for *liikkua*, a special treatment is required, as the verb can receive either an agent or a patient as its subject, but both arguments cannot appear with the same verb occurrence.

Our solution is to create a single frameset for verbs like *liikkua*: one with both Arg0 and Arg1, as with the transitive verb *liikuttaa*. Only with *liikkua*, it is explicitly marked in the frameset that these two arguments are

liikkua.1: to move, be moved		liikuttaa.1: to move something	
Arg0	Entity moving actively	Arg0	Entity moving arg1
Arg1	Entity whose movement something causes **if not Arg0**	Arg1	Entity moved
Arg2	Place	Arg2	Place

Figure 3.6: Framesets of the Finnish verbs with the meaning *to move*. Left: the intransitive verb that takes as its subject either an agent or a patient. Right: the transitive causative verb for moving. *[Figure adapted from Paper VII.]*

mutually exclusive, that is, only one of them should be annotated with any given verb occurrence. For an illustration of the framesets of *liikkua* and *liikuttaa*, see Figure 3.6.

One final note applies to the frameset creation in general. In order to make the frameset creation process as fluent as possible, we use a *batch* system, where the same frameset can be given to multiple verbs at the same time. Consider, for instance, different verbs of affection. If an annotator is creating a frameset for the verb *to like*, the batch system enables the annotator to assign the same frameset to other verbs that have the same arguments and argument descriptions: for example *to love*, *to care* or *to adore* could be candidates for such verbs. It should be noted that we require the argument descriptions to be suitable for all verbs receiving framesets in the same batch. This means that even though verbs of dislike have the same numbered arguments as the verbs of affection, they should not be given the same framesets, as the argument descriptions are unsuitable. In addition to the batch system, it is also possible to reuse framesets by copying them from previously framed verbs to new ones.

3.3.3 Annotation and its evaluation

Similarly to the work of Paper II, the work on the general Finnish PropBank utilizes an automatic morphological analyzer to find all possible verbs from the underlying treebank. In the case of the general Finnish PropBank the tool used is OMorFi, which became available between the two PropBank efforts. All tokens that receive a verbal reading (in OMorFi analyses, unlike in FinCG analyses, also participles receive the main POS tag *V*) or a reading indicating a *minen*-derivation,[2] are annotated. According to OMorFi, TDT contains 49,727 possible verb tokens, which belong to 2,946 different possible lemmas.

The framing and annotation was done by a total of six different annotators, using a mixture of single and double annotation. The annotation

[2]resembles the English *ing*-participle and similarly to it, can be analyzed as either a noun or a verb depending partly on context

was started with full double annotation, in order to ensure the best possible annotation quality even in the initial learning phase. Afterwards, we moved towards single annotation in a controlled manner: verbs with a large number of occurrences were partially double annotated, meaning that some occurrences were double annotated and some were left to be single annotated, and verbs with only few occurrences were single annotated. At this point in time, no phase of consistency checks has been performed after the disagreements of the annotators have been solved, but it seems likely that such a phase would be beneficial, both to ensure the consistency of the PropBank annotation and to correct syntax annotation errors that have been uncovered in this new round of annotation.

In addition to partly using single annotation, the annotation protocol of the Finnish PropBank differs from that of TDT for one important reason: before the actual annotation can begin, the annotators must first have an initial set of framesets agreed on. If necessary, the framesets can be revised in the course of the annotation to fit the examples in the corpus better, and new framesets can also be created after the annotation has started. Also unlike in TDT, no large-scale pre-processing is used, although certain ArgMs that always occur when a certain syntactic dependency type is present (such as *ArgM-MOD* with the auxiliary type *aux*) are automatically pre-filled. The annotation protocol of the Finnish PropBank is illustrated in Figure 3.7.

At the time of the writing of Paper VII, only a very limited portion of the general Finnish PropBank was completed: approximately 11.4% of lemmas and 18.2% of tokens had been annotated. Regardless, the paper presented a preliminary evaluation of the annotator accuracy using F_1-score. The detailed accuracy evaluation from Paper VII is given in Table 3.1.

This Table shows that the overall annotator accuracy across all annotators and all different arguments was 94.1%, which demonstrates high annotation quality. Further, it can be seen that especially the accuracy on the numbered arguments is high, compared to the ArgMs as well as the separate evaluation of the external arguments. This is an expected result, seeing that also Palmer et al. (2005) reported results showing that numbered arguments were easier to annotate than ArgMs in the original PropBank. As for the external arguments, it is intuitive that they should be more difficult than numbered arguments in general: due to their position outside the immediate dependents of the verb, they are easier to overlook, and in addition, the annotator is required to identify the correct argument head word, which is not the case for other arguments.

At the time of finalizing this thesis, in late May 2014, the PropBank annotation is complete, with all verbs and all of their arguments annotated, and the manuscript describing the final resource is under preparation. The final overall annotator accuracy of all double annotated occurrences was 91.7% in F_1-score, indicating that the annotation quality remains high.

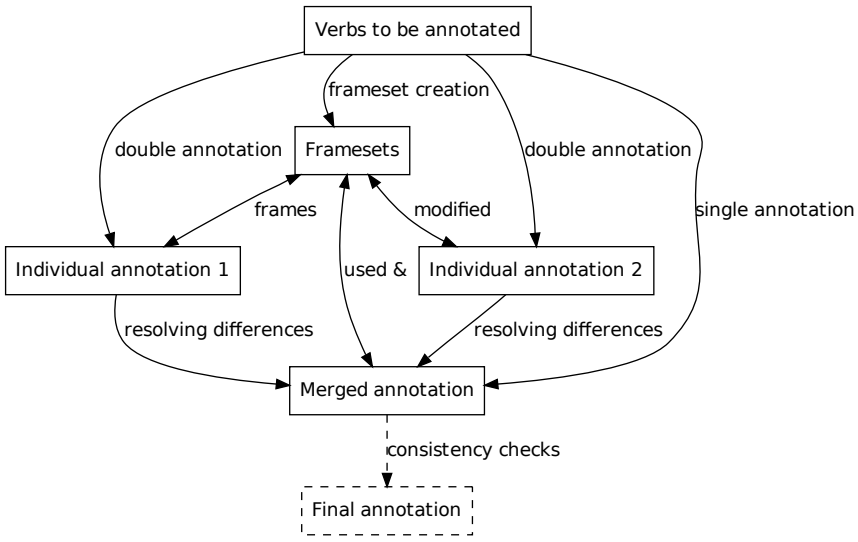

Figure 3.7: The annotation workflow of the Finnish PropBank. The verb occurrences in the corpus are first used to create an initial set of framesets, which are then used for the annotation, which can be single or double annotation. If necessary, the framesets can be revised or new framesets can be created during annotation. Afterwards, an additional round of consistency checks is envisioned.

	Ann. 1	Ann. 2	Ann. 3	Ann. 4	Ann. 5	Ann. 6	All
Numbered (n=29,076)							
Recall	98.1	96.5	96.5	94.9	97.8	95.6	96.9
Precision	98.5	98.0	98.0	95.1	98.1	94.5	97.4
F-score	98.3	97.2	97.3	95.0	97.9	95.1	97.1
ArgM (n=15,771)							
Recall	92.5	86.6	87.3	83.7	90.1	82.8	87.8
Precision	92.9	87.3	86.6	85.2	92.6	82.0	88.2
F-score	92.7	86.9	87.0	84.4	91.3	82.4	88.0
xarg (n=3,118)							
Recall	93.3	80.8	79.3	70.3	87.4	85.9	86.0
Precision	97.8	97.8	92.3	70.3	94.7	84.3	92.7
F-score	95.5	88.5	85.3	70.3	90.9	85.1	89.2
overall (n=44,847)							
Recall	96.3	93.0	93.4	90.9	95.2	91.6	93.9
Precision	96.7	94.2	94.1	91.5	96.3	90.6	94.3
F-score	96.5	93.6	93.7	91.2	95.8	91.1	94.1

Table 3.1: Annotator accuracy results per annotator, both overall and separately for numbered arguments and ArgMs. Also a separate evaluation of the external arguments (*xarg*) is given. Note that for the F_1-scores the external arguments are also included in the counts of numbered arguments and ArgMs, seeing that each external argument is also one of these two argument types. *[Table from Paper VII.]*

Chapter 4

Conclusions

This thesis has described seven studies in the area of corpus development in the domains of clinical and general Finnish, with the overall goal of enabling and advancing Finnish NLP by providing the essential resources for the purposes of parsing and semantic role labeling. The studies of the thesis related to four issues that were defined as research objectives in the introduction of the thesis.

The first and most important objective of this work was to develop a freely available treebank and statistical parser for Finnish. Both of these are important resources that did not previously exist for Finnish, which has seriously hindered Finnish NLP research. This objective was addressed in two parts. First, Paper I presented a small-scale treebank for *ICU Finnish*, the specific language of patient reports in a Finnish intensive care unit. Paper II further extended this treebank and made it publicly available. Second, Paper III presented the idea and the very first version of the first freely available treebank of general Finnish, the Turku Dependency Treebank (TDT). This treebank was later extended in Papers IV and V, and the current, full version of the treebank was presented in Paper VI, which addressed the objective for general Finnish. Also, in Papers I and VI statistical parsers for the two languages were presented, as was stated in the first objective.

The second objective was to further enhance the corpora with annotation suitable for semantic role labeling. Paper II explored this issue by describing the PropBank annotation of the ICU Finnish corpus. For general Finnish, this objective was addressed although not completely achieved in the context of this thesis: Paper VII described the first steps towards a PropBank annotated on top of TDT, and at the time of finalizing this thesis, the PropBank annotation work is finished and the related manuscript is under preparation.

The third objective of this research considered annotation schemes. Our aim was to find whether building a completely new annotation scheme for

syntax annotation of Finnish, and likewise for semantic role labeling, could be avoided. We can now answer this question in the affirmative.

In the work of Paper I we found that only minor modifications were needed for the Stanford Dependency (SD) scheme to be suitable for ICU Finnish. The scheme developed in this work was further refined in the development of the Turku Dependency Treebank and found to be suitable for general Finnish as well; intermediate versions of the work and the scheme are described in Papers III and IV, and the final treebank and its scheme are discussed in Paper VI. With respect to the Finnish-specific SD scheme, we also find that although it has been developed fully independently of the original, English SD scheme, both schemes have developed in the same direction, indicating that the Finnish-specific scheme is in line with the original scheme designers' intentions. Paper V also contributes towards the third objective with regard to syntax, by identifying phenomena difficult for human annotators in this particular scheme.

As for the annotation of semantic roles, Paper II shows that the PropBank scheme is suitable for ICU Finnish and that it is compatible with the SD scheme as well. For general Finnish, Paper VII shows that almost exactly the same scheme could be employed in the semantic role annotation, with only minor changes due to issues not faced in the limited domain of clinical language.

Thus, we find that for both tasks, there was no need to construct a custom annotation scheme from scratch, but rather a scheme developed for English could be adapted for Finnish. Using an existing scheme is beneficial not only due to the reduced effort, but also because such a scheme makes the resource developed compatible with other resources using it. This is especially true of the SD and PropBank schemes, as both have been popular across several languages.

Finally, the fourth objective of the work was to find a suitable annotation workflow or process for the individual studies of this thesis. The studies applied slightly different approaches, partly due to their different scales: both the clinical treebank and TDT were double annotated with a merging phase, whereas the clinical PropBank was annotated collaboratively and the Finnish PropBank applied a mixture of double and single annotation. Table 4.1 yet summarizes the main features of annotation workflows applied in the different annotation projects of the work.

All of these different annotation protocols were applied with the ideal of producing as good annotations as possible in the given time, which differed between the projects. In the syntax annotation, full double annotation was used in order to settle the scheme and to maximize the quality of the annotation. In the case of the PropBanks, single or partial single annotation were justified based on time constraints. The clinical PropBank was annotated collaboratively, as it was a pilot project. In the Finnish PropBank,

Project	Double annotation?	Technical aid?	Annotators?
Clinical Treebank	full	none	2
Clinical PropBank	none (collaborative)	none	4
TDT (base)	full	parser	5
TDT (2nd layer)	full	pre-processing	6
Finnish PropBank	partial	(pre-processing)	6

Table 4.1: The different workflows used in the annotation projects of the thesis.

the bigger of the two PropBanks, a portion of the verbs were first double annotated, and as the evaluation showed encouraging results, the project moved towards single annotation to increase the speed of the work.

In the general Finnish work, also technical aids were used where applicable. In the *base* layer of TDT, a preliminary parser was used to pre-annotate data, and in the *conjunct propagation and additional dependencies layer* as well as in the Finnish PropBank, a pre-processing software was used to make suggestions to the annotators. The pre-annotation protocol of the *base* syntax layer was studied in Paper IV, and according to its results, the protocol could be highly useful to a beginning annotator with respect to both speed and accuracy, and while a similar benefit was not observed for more experienced annotators, it was not clearly harmful to them, either.

Paper V contributes to the fourth objective especially, as it studies the syntax annotation process itself. It reports the most difficult phenomena to annotate in the particular annotation scheme and points towards ways in which the annotation software could be improved. For future annotation efforts in a compromise setting, it also suggests a method that can detect sentences likely containing annotation errors so that these sentences can be offered for inspection.

Overall, the clinical language studies of this work have preceded those on general Finnish, and as such, have also served as pilot studies for them. Based on these studies, we were encouraged to attempt similar projects on a larger scale. The clinical annotation projects also enabled us, on the levels of both syntax and semantic roles, to preliminarily estimate the suitability of the annotation schemes for Finnish as well as to envision annotation protocols to fit the larger scale general Finnish tasks. Interestingly, although it could be said that the clinical treebank project has helped in the construction of TDT, in defining the scheme among other things, it does not seem that the general language data in turn is very helpful in parsing ICU Finnish. According to the results of Laippala et al. (2013), using TDT as training data does not substantially increase parsing performance. In fact, when used on top of a body of other clinical data, TDT even seems to cause a minor performance decrease.

As a result of this research, a substantial amount of Finnish language technology research previously impossible has now become feasible. By enabling statistical parsing of the language, the work has opened a path towards further, more advanced applications, such as question answering, information extraction, text understanding, text generation, machine translation — or indeed any application that can benefit from parsing. The work of this thesis has already enabled further research, as exemplified by the parsing study of Bohnet et al. (2013), the parsebank of Ginter et al. (2013), the machine translation project in collaboration with Convertus AB and the new, universal version of SD (de Marneffe et al., 2014) partly inspired by existing SD treebanks for different languages, including TDT. This research, in turn, has given the Finnish language attention from the international NLP community especially from the point of view of parsing, attention that it has previously not received for an extended period of time, due to the lack of resources.

As always is the case in research, there is more work to be done. Some examples of future work directions include at least the following. Although Paper VI presents a method for retrieving disambiguated morphological analyses of good quality for TDT, the morphology of the treebank can further be improved and indeed after the publication of the current treebank version, there has been an effort to manually annotate morphological analyses (currently still unpublished). In addition, it would perhaps also be possible to apply a method similar to that presented in Paper V for syntax to subject the most likely erroneous morphological analyses to inspection.

The accuracy of the baseline parser of Finnish presented in Paper VI can also be further improved, and in fact, at the time of writing this thesis, it already has been, by Bohnet et al. (2013). One possible method is to simply increase the size of TDT with single annotation; as shown by Dligach et al. (2010), when solely training material for a machine learning method is desired, single annotating more data is preferable to double annotating less data. Also related to parsing, in addition to the parsebank briefly described in Section 2.3.7, another parsebank of Finnish, of larger scale, is currently under development in the research group, and the first results of this project are already becoming available (Kanerva et al., 2014; Kanerva and Ginter, 2014).

As for PropBanking Finnish, the complete PropBank is finished and currently awaiting publication, and the same manuscript will also describe a semantic role labeling tool providing a baseline on the task for Finnish. One potential future work direction here is attempting to further improve the performance of the tool using statistical methods. Additionally, the PropBank itself could be enhanced with annotation of noun argument annotation, that is, a NomBank (Meyers et al., 2004a,b), or modified along the lines suggested by Wanner et al. (2012) to support text generation.

Bibliography

Babko-Malaya, O. (2005a). Guidelines for PropBank framers. Technical report, University of Colorado Boulder.

Babko-Malaya, O. (2005b). PropBank annotation guidelines. Technical report, University of Colorado Boulder.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of COLING-ACL'98*, pages 86–90.

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

Bloomfield, L. (1933). *Language*. The University of Chicago Press.

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

Bohnet, B. and Nivre, J. (2012). A transition-based system for joint part-of-speech tagging and labeled non-projective dependency parsing. In *Proceedings of EMNLP-CoNLL'12*, pages 1455–1465.

Bohnet, B., Nivre, J., Boguslavsky, I., Farkas, R., Ginter, F., and Hajič, J. (2013). Joint morphological and syntactic analysis for richly inflected languages. *Transactions of the Association for Computational Linguistics*, 1(October):429–440.

Carroll, J., Minnen, G., and Briscoe, T. (1999). Corpus annotation for parser evaluation. In *Proceedings of LINC'99*, pages 35–41.

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

Choi, J. and Palmer, M. (2010). Retrieving correct semantic boundaries in dependency structure. In *Proceedings of LAW IV*, pages 91–99.

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

Chomsky, N. (1957). *Syntactic structures*. Mouton.

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

Dang, H. T., Kipper, K., Palmer, M., and Rosenzweig, J. (1998). Investigating regular sense extensions based on intersective Levin classes. In *Proceedings of COLING-ACL'98*, pages 293–299.

Dligach, D., Nielsen, R., and Palmer, M. (2010). To annotate more accurately or to annotate more. In *Proceedings of LAW IV*, pages 64–72.

Duran, M. S. and Aluísio, S. M. (2011). PropBank-Br: a Brazilian treebank annotated with semantic role labels. In *Proceedings of STIL'11*, pages 1862–1867.

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

Fort, K. and Sagot, B. (2010). Influence of pre-annotation on POS-tagged corpus development. In *Proceedings of LAW IV*, pages 56–63.

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

Ginter, F., Nyblom, J., Laippala, V., Kohonen, S., Haverinen, K., Vihjanen, S., and Salakoski, T. (2013). Building a large automatically parsed corpus of Finnish. In *Proceedings of NODALIDA'13*, pages 291–300. Linköping University Electronic Press.

Hajič, J. (1998). Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová*, pages 106–132. Karolinum, Charles University Press, Prague, Czech Republic.

Hajič, J., Ciaramita, M., Johansson, R., Kawahara, D., Martí, M. A., Màrquez, L., Meyers, A., Nivre, J., Padó, S., Štěpánek, J., Straňák, P., Surdeanu, M., Xue, N., and Zhang, Y. (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., and Alho, I. (2004). *Iso suomen kielioppi / Grammar of Finnish*. Suomalaisen kirjallisuuden seura.

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

Haverinen, K. (2012). Syntax annotation guidelines for the Turku Dependency Treebank. Technical Report 1034, Turku Centre for Computer Science. Second Edition, Revised for the treebank release of July 2013.

Haverinen, K., Ginter, F., Pyysalo, S., and Salakoski, T. (2008). Accurate conversion of dependency parses: Targeting the Stanford scheme. In *Proceedings of the Third International Symposium on Semantic Mining in Biomedicine (SMBM 2008), Turku, Finland*, pages 133–136.

Jurafsky, D. and Martin, J. (2009). *Speech and Language Processing — An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Pearson Education.

Kanerva, J. and Ginter, F. (2014). Post-hoc manipulations of vector space models with application to semantic role labeling. In *Proceedings of the 2nd Workshop on Continuous Vector Space Models and their Compositionality (CVSC)@ EACL*, pages 1–10.

Kanerva, J., Luotolahti, J., Laippala, V., and Ginter, F. (2014). Syntactic n-gram collection from a large-scale corpus of internet finnish. In *Proceedings of the Sixth International Conference Baltic HLT 2014*.

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

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

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

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

Laippala, V., Ginter, F., Pyysalo, S., and Salakoski, T. (2009). Towards automated processing of clinical Finnish: A sublanguage analysis and a rule-based parser. *International Journal of Medical Informatics, Special Issue on Mining of Clinical and Biomedical Text and Data*, 78(12):e7–e12.

Laippala, V., Viljanen, T., Airola, A., Nyblom, J., Salanterä, S., Salakoski, T., and Ginter, F. (2013). Statistical parsing of varieties of clinical Finnish. In *Proceedings of Louhi'13*, pages 1–6.

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

Levin, B. and Hovav, M. R. (1994). *Unaccusativity: At the syntax–lexical semantics interface*, volume 26 of *Linguistic Inquiry*. MIT Press.

Lin, D. (1998). A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114.

Lindén, K., Silfverberg, M., and 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*, volume 41 of *Communications in Computer and Information Science*, pages 28–47.

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

de Marneffe, M.-C., Connor, M., Silveira, N., Bowman, S. R., Dozat, T., and Manning, C. D. (2013). More constructions, more genres: Extending Stanford Dependencies. In *Proceedings of Depling'13*, pages 187–196.

de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal stanford dependencies: a cross-linguistic typology. In *Proceedings of LREC'14*, pages 4585–4592.

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

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

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

McDonald, R., Nivre, J., Quirmbach-Brundage, Y., Goldberg, Y., Das, D., Ganchev, K., Hall, K., Petrov, S., Zhang, H., Täckström, O., Bedini, C., Bertomeu Castelló, N., and Lee, J. (2013). Universal dependency annotation for multilingual parsing. In *Proceedings ACL'13*, pages 92–97.

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

Mel'čuk, I. (1979). *Studies in dependency syntax*. Karoma Publishers.

Mel'čuk, I. (1988). *Dependency Syntax: Theory and Practice*. State University of New York Press.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004a). Annotating noun argument structure for NomBank. In *Proceedings of LREC-2004*.

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004b). The NomBank project: An interim report. In *Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

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

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

Nivre, J. (2006). *Inductive Dependency Parsing*, volume 34 of *Text, Speech and Language Technology*. Springer.

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

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

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

Nyblom, J., Kohonen, S., Haverinen, K., Salakoski, T., and Ginter, F. (2013). Predicting conjunct propagation and other extended Stanford Dependencies. In *Proceedings of Depling'13*, pages 252–261.

Palmer, M., Bhatt, R., Narasimhan, B., Rambow, O., Sharma, D. M., and Xia, F. (2009). Hindi syntax: annotating dependency, lexical predicate–argument structure, and phrase structure. In *Proceedings of ICON'09*.

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

Palmer, M., Gildea, D., and Xue, N. (2010). *Semantic Role Labeling*. Morgan & Claypool Publishers.

Perlmutter, D. (1978). Impersonal passives and the unaccusative hypothesis. In *Proceedings of the Fourth Annual Meeting of the Berkeley Linguistic Society*, pages 157–189.

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

Pyysalo, S., Ginter, F., Laippala, V., Haverinen, K., Heimonen, J., and Salakoski, T. (2007). On the unification of syntactic annotations under the Stanford Dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP'07*, pages 25–32. Association for Computational Linguistics (ACL).

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

Rehbein, I., Ruppenhofer, J., and Sporleder, C. (2009). Assessing the benefits of partial automatic pre-labeling for frame-semantic annotation. In *Proceedings of LAW III*, pages 19–26.

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

Schneider, G., Rinaldi, F., and Dowdall, J. (2004). Fast, deep-linguistic statistical dependency parsing. In *Proceedings of COLING'04 Workshop on Recent Advances in Dependency Grammar*, pages 33–40.

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

Silfverberg, M. and Lindén, K. (2010). Part-of-speech tagging using parallel weighted finite-state transducers. In *Proceedings of IceTAL'10*, pages 369–380.

Silveira, N., Dozat, T., de Marneffe, M. C., Bowman, S., Connor, M., Bauer, J., and Manning, C. D. (2014). A gold standard dependency corpus for english. In *Proceedings of LREC'14*, pages 2897–2904.

Sleator, D. D. and Temperley, D. (1993). Parsing English with a Link Grammar. In *Proceedings of the Third International Workshop on Parsing Technologies*, pages 277–291.

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

Surdeanu, M., Johansson, R., Meyers, A., Màrquez, L., and Nivre, J. (2008). The CoNLL-2008 shared task on joint parsing of syntactic and semantic dependencies. In *Proceedings of CoNLL'08*, pages 159–177.

Tesniére, L. (1959). *Éléments de syntaxe structurale*. Éditions Klinksieck.

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

Valkonen, K., Jäppinen, H., and Lehtola, A. (1987). Blackboard-based dependency parsing. In *Proceedings of IJCAI'87 - Volume 2*, pages 700–702.

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

Wanner, L., Mille, S., and Bohnet, B. (2012). Towards a surface realization-oriented corpus annotation. In *Proceedings of INLG '12*, pages 22–30.

Xue, N. and Palmer, M. (2009). Adding semantic roles to the Chinese treebank. *Natural Language Engineering*, 15(Special issue 01):143–172.

Zaghouani, W., Diab, M., Mansouri, A., Pradhan, S., and Palmer, M. (2010). The revised Arabic PropBank. In *Proceedings of LAW IV*, pages 222–226.

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

Publication Reprints

Paper I

Parsing clinical Finnish: Experiments with rule-based and statistical dependency parsers.

Haverinen, K., Ginter, F., Laippala, V., and Salakoski, T. (2009). In *Proceedings of NODALIDA'09, Odense, Denmark*, pages 65–72.

Parsing Clinical Finnish: Experiments with Rule-Based and Statistical Dependency Parsers

Katri Haverinen,[1] **Filip Ginter,**[1] **Veronika Laippala,**[1,2] **and Tapio Salakoski**[1]
[1]Department of Information Technology
[2]Department of French Studies
University of Turku
Joukahaisenkatu 3-5
20520 Turku, Finland
first.last@utu.fi

Abstract

In this paper, we present a new syntactically annotated corpus consisting of daily notes from an intensive care unit in a Finnish hospital. Using the corpus, we perform experiments with both rule-based and statistical parsers. We apply an existing rule-based parser specifically developed for this clinical language and create a set of conversion rules for transforming the constituency scheme of this parser into the dependency scheme of the corpus. The statistical parser is induced from the corpus using the MaltParser system.

We find that even with a modestly-sized corpus, the statistical parser achieves results comparable to those previously reported on a number of languages using considerably larger corpora. The accurate constituency-to-dependency conversion improves the applicability of the rule-based parser by inferring grammatical roles, thus deepening its analyses.

1 Introduction

The potential advantages of applying natural language processing methods in the clinical domain are numerous, with many useful applications in decision support, patient management and profiling, and mining trends (see, e.g., the recent review by Friedman and Johnson (2006)). While certain applications, such as document retrieval and trend mining, can solely rely on word frequency-based statistical methods, a number of applications build on a detailed analysis of the text, typically involving syntactic parsing.

In this paper, we describe experiments on full parsing of Finnish intensive care unit (ICU) nursing documents written in a specific language referred to as ICU Finnish throughout the paper. The main contributions of this work are a corpus of ICU Finnish, syntactically annotated in an adapted version of the Stanford dependency (SD) scheme, and both rule-based and statistical parsing experiments on this corpus. We apply the rule-based parser of Laippala et al. (2009) developed for ICU Finnish, and develop a conversion from its native constituency scheme to the SD scheme. We also conduct experiments with a statistical parser induced from the ICU Finnish corpus using the MaltParser (Nivre et al., 2007) system. This allows us to evaluate and contrast the relative advantages of the two parsing approaches in this domain.

2 Related work

There are numerous applications of full syntactic parsers in the clinical domain. For instance, the Stanford parser has been applied to the extraction of noun phrases with full phrase structures and to negation detection in clinical radiology reports (Huang and Lowe, 2007; Huang et al., 2005). There have also been many studies on the adaptation of existing parsers to the specific domain of biomedical language. For example, Szolovits (2003) describes a method for expanding the Link Grammar (LG) lexicon with UMLS Specialist lexicon terms to improve its applicability to medical texts and Pyysalo et al. (2006) incorporate into LG a domain-adapted part-of-speech tagger.

The different ways to represent natural language syntax can be broadly distinguished into two categories. A constituency analysis divides the sentence into nested phrases, whereas a dependency analysis consists of a set of labelled dependencies between pairs of words. In this work, we focus on dependency parsing because of its benefits in applications and parser evaluation (see for example Lin (1998), Clegg and Shepherd (2007), and Nivre (2008b)), as well as its applicability to languages with a relatively free word order, such

Kristiina Jokinen and Eckhard Bick (Eds.)
NODALIDA 2009 Conference Proceedings, pp. 65–72

Yövuoro	Nightshift
Potilas levoton, valittaa kipua.	Patient restless, complains of pain.
Annettu 100mg [lääke] hieman rauhottui.	Given 100mg [drugname] a little calmed down.
HENGITS: Hapettuu hyvin repiraattorissa.	BREATING: Oxidates well in repirator.
Putkesta hiukan nest. illalla.	A little liq. from the drain in the evening.
Diureesi: riittävää.	Diuresis: sufficient.
Hemodyn: annettu 50 mg/h [lääke],	Hemodyn: given 50 mg/h [drugname],
heikohko vaste vaihdettu [lääke].	rather weak response changed to [drugname].
OMAISET: vaimo soittanut jutellut lääkärin kanssa.	RELATIVES: wife called talked to doctor.

Figure 1: Example of ICU Finnish (left column) and its exact translation (right column), including spelling errors, capitalization, and the like.

as Finnish. We apply the Stanford dependency scheme (de Marneffe et al., 2006; de Marneffe and Manning, 2008), which has recently been employed in several studies especially in the biomedical domain, but also in other contexts. For an extensive list of applications, we refer to the review by de Marneffe and Manning (2008).

While numerous corpora and parsers exist for English and many other languages, resources for Finnish are scarce. For instance, there is no publicly available syntactically annotated corpus suitable for statistical parser induction. The only publicly available full parser is Connexor Machinese Syntax,[1] a closed-source commercial dependency parser for the general language. Other tools include FinTWOL and FinCG,[2] a morphological analyzer and a Constraint Grammar parser that resolves morphological ambiguity (Koskenniemi, 1983; Karlsson, 1990). The rule-based parser of Laippala et al. (2009) used in this work was developed for the clinical domain, and builds full constituency analyses on top of the morpholexical analyses provided by FinTWOL and FinCG.

3 ICU Finnish in the Stanford dependency scheme

ICU Finnish differs from standard Finnish in many ways (for details, see the discussion by Laippala et al. (2009)). Some of the most distinguishing features present in ICU Finnish, as well as many clinical sublanguages, are frequent misspellings, abbreviations and technical terms, telegraphic sentences, syntactic structures that would not be allowed in standard language, and frequent omissions of main verbs, subjects and copulas. Figure 1 is an illustration of ICU Finnish.[3] The effects of ICU Finnish features on analyzing the syntax will be more thoroughly discussed in Section 3.2.

3.1 The SD scheme

In the SD scheme, the syntactic structure of a sentence is represented as a directed graph where the nodes correspond to words and the edges correspond to dependencies. Unlike in most dependency schemes, SD graphs are not necessarily trees and may even contain directed cycles. Each dependency is labelled with a dependency type that represents the syntactic function of the dependent word. In the latest version of the SD scheme (de Marneffe and Manning, 2008), there are in total 55 dependency types.

We have chosen the SD scheme due to its numerous successful applications in different contexts. Further, de Marneffe and Manning find the scheme applicable in parser comparison. This particular aspect of the scheme is of importance with respect to this work, as one part of this study is a comparison of two parsers. Alternative schemes, such as Grammatical Relations (Carroll et al., 1998) and the Connexor Machinese Syntax scheme, were also considered. The former has been suggested by its authors to be suitable for multiple languages, and the latter is a scheme designed for standard Finnish.

3.2 Applying the SD scheme to ICU Finnish

The SD scheme was designed for standard English. In this section, we describe the modifications made in order to adapt it to ICU Finnish. These modifications include both those that are required by Finnish in general, and those implied by the nature of the ICU sublanguage. For an illustration of the modified SD scheme, see Figure 2. As a detailed description of the SD scheme is beyond the scope of this paper, we only discuss our modifications to it and refer to the description by de Marneffe and Manning (2008).

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

[2] http://www.lingsoft.fi

[3] Due to the confidential nature of the patient data, these, as well as all examples used in this paper, are not actual sentences from the data, but rather illustrative examples.

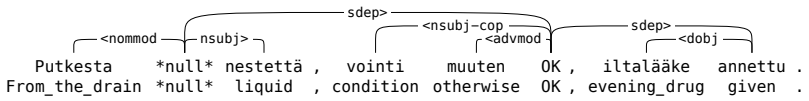

Figure 2: The modified SD scheme. Note the following features: nominal modifiers (*nommod* dependencies), dependencies between sentences (*sdep*), null verbs that represent omitted main verbs, explicit marking of copula subjects (*nsubj-cop*), and the use of direct object (*dobj*) in passive sentences. The sentence can be roughly translated as *Liquid from the drain, condition otherwise OK, evening drug given.*

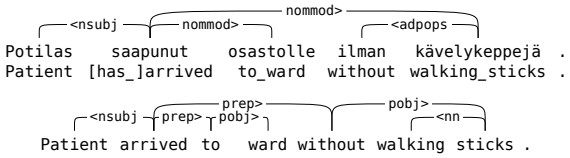

Figure 3: Top: usage of the new dependency types *nommod* and *adpos*. Bottom: the corresponding English sentence and annotation in the SD scheme. Note that the type *nommod* is used both for nominal inflection and prepositional phrases.

3.2.1 Prepositional phrases

In the Finnish language, prepositions are relatively rare. Most English clauses with prepositional phrases have Finnish equivalents that use nominal inflection. For an example of a typical case, see Figure 2.

Seeing that inflectional and prepositional structures are semantically similar, it would be desirable to represent them in a similar manner also in the dependency structure. Therefore, we introduce a new dependency type, *nommod* (*nominal modifier*), to represent inflectional structures. This same type can also be used in sentences with actual pre- and postpositions. Only one additional type is needed for prepositional structures, a type named *adpos* (*adposition*). For an illustration of the usage of these two types, see Figure 3. The structure given to prepositional phrases is similar to that used in the scheme of the Pro3Gres parser (Schneider et al., 2004).

3.2.2 Passive subjects

Certain Finnish clause types, contrary to their English counterparts, do not require a subject. One that has a particular effect on our work is the passive voice. The surface subject in English passive clauses corresponds to both surface and deep object in Finnish. Therefore, we have not used the dependency type *nsubjpass* at all, and have used *dobj* instead.

3.2.3 Dependencies between sentences

A third modification to the SD scheme is required by the nature of the ICU language: sentence boundaries are often not clearly marked, or they lack punctuation altogether (see Figure 4). We split the text into separate sentences only when there is explicit punctuation that marks the sentence boundary. Recovering sentence boundaries that have no explicit surface marking is left to the parser, as recognizing them would be difficult for standard sentence splitters that lack syntax information. We have thus introduced a new dependency type, *sdep*, to connect these isolated sentences that are not explicitly coordinated or subordinated. To produce an analysis that is æsthetic from a scheme design point of view, if several *sdep* dependencies are needed in the same surface sentence, they are chained. This is to avoid unnecessarily long dependencies that are difficult for parsers to recover.

3.2.4 Omissions

In ICU Finnish, a frequent syntactic feature that has a notable effect on parsing the language is the omission of different sentence elements. One example of this is the omission of copulas and auxiliaries, which have little effect on sentence semantics. Consider, for example, *The patient is awake* vs. *The patient awake*.

In some cases, it is even possible to omit the main verb of a sentence. For instance, the structure

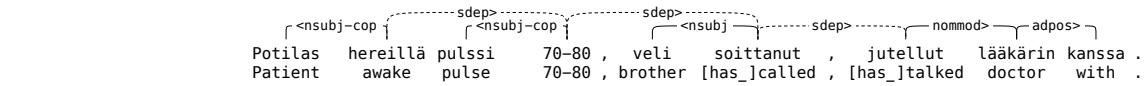

Figure 4: The purpose of the *sdep* dependencies is to combine the independent sentences under one surface sentence into a single analysis. Without the dashed *sdep* dependencies, the analysis would contain separate islands. This sentence can be roughly translated as *Patient awake pulse 70-80, brother called, talked with the doctor*.

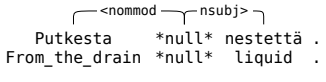

Figure 5: Missing main verbs are represented by a null verb, in order to construct a dependency analysis for sentences such as this. The sentence can be roughly translated as *Liquid from the drain*.

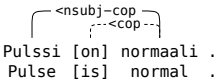

Figure 6: The new dependency type *nsubj-cop*, used instead of *nsubj* in copula clauses. Note that the analysis stays essentially the same, regardless of the presence or absence of the copula.

Putkesta nestettä (*Liquid from the drain*) is common in ICU Finnish, though it would be judged fragmentary in standard Finnish. Here, the case of the noun *putkesta* (*from the drain*) expresses the direction of the liquid, and the actual verb (*to come*) can therefore be omitted, as its meaning is clear in the context. This poses a problem for most dependency schemes, as the main verb of a clause is also its head word. To be able to analyze the sentences with a missing main verb (21% of the sentences in the corpus), we have manually introduced a *null verb* in those sentences to represent the missing verb. See Figure 5 for an illustration of this solution.

Because the purpose of the null verb is to represent a word that is absolutely necessary for the construction of an SD analysis, null verbs are introduced only when the main verb is omitted. Copulas and auxiliaries never act as governors in the SD scheme and thus do not require a null verb to be inserted.

Finally, the frequent omissions of copulas require another minor modification to the SD scheme, the introduction of the dependency type *nsubj-cop*. The *nsubj* type used in the original SD scheme for both standard and copula subjects is in our version of the scheme replaced by *nsubj-cop* in copula clauses. This is to differentiate the special case of copula subjects, where, in the SD scheme, the governor of the dependency is not a verb but, for example, an adjective. For an illustration of the use of *nsubj-cop*, see Figure 6.

4 Performance measures

When evaluating the quality of our corpus, as well as the performance of the parsers in the experiments described below, we use the following measures.

Precision (P) is defined as the proportion of dependencies in the parser output that are also present in the gold standard. *Recall (R)*, in turn, is the proportion of dependencies in the gold standard that are also present in the parser output. These two are combined into an F*-score*, defined as $F = \frac{2PR}{P+R}$.

Labelled attachment score (A_L) is the proportion of tokens that are assigned the correct head and dependency label according to the gold standard, and *unlabelled attachment score* (A_U) is the proportion of tokens that are assigned the correct head, regardless of the dependency label (Nivre, 2008a). Note that A_L and A_U are defined for tree structures where each token has exactly one head. As noted previously, analyses in the SD scheme are not necessarily trees, and thus the two measures are not directly applicable to it.

5 Corpus annotation and statistics

As one of the primary contributions of this work, we have annotated a corpus of 1019 ICU Finnish sentences with 7614 tokens of which 6082 are non-punctuation. The text of the corpus consists of notes written by nurses about the condition of a patient, often with respect to standard topics such as breathing, hemodynamics, diuresis and relatives.

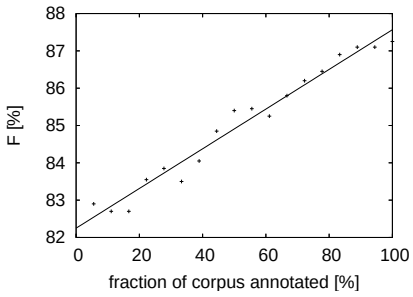

Figure 7: Inter-annotator agreement in F-score at various stages of the corpus annotation with a trend line. Note that the A_L and A_U measures are not reported, as the SD analyses are not necessarily trees.

The corpus currently consists of sentences from four different patient reports, as we decided to annotate full reports rather than randomly selected individual sentences, to enable further research, for example in report summarization.

The dependency annotation has in total 5194 dependencies. Only 2.9% of all sentences and 0.5% of all tokens are non-projective. The effect of non-projectivity on parsing ICU Finnish is thus negligible.

We used full double annotation, that is, each sentence was independently annotated by two annotators, and disagreements were jointly resolved. To evaluate the quality of the corpus, we measured inter-annotator agreement, defined as the average of the agreements of the two annotators against the final annotation. The average inter-annotator agreement on the whole corpus was 87.25% F-score. Figure 7 illustrates the growth of the inter-annotator agreement as the annotators become familiar with the task and the scheme.

We estimate that the current corpus has taken 70 man-hours of annotation work to develop, including both the independent annotation work by individual annotators and the joint resolving of disagreements. The disagreement resolving took in total approximately 30 man-hours. We used a custom software for annotation and disagreement resolution.

6 Experiments on the corpus

In this section, we discuss the experiments that the newly built corpus has enabled us to perform. We first describe our experiments on the rule-based approach, including the conversion rules required for the evaluation of the parser. We then present results of another experiment, which uses a statistical approach.

In order to be able to use the A_L and A_U performance measures described in Section 4, as well as to maintain comparability of results with MaltParser which produces tree analyses, the treeness of all analyses in all experiments was assured by breaking the possible cycles present in the gold standard. Punctuation tokens were excluded from all performance measurements and the null verbs representing omitted verbs were preserved in the parser input.

6.1 Parsing experiments with a rule-based parser

As the first part of our experiments, we apply the rule-based parser of Laippala et al. (2009) whose reported coverage is up to 75% of ICU Finnish sentences with an oracle best parse performance of above 90% in terms of the PARSEVAL metric (Black et al., 1991).

6.1.1 The dependency conversion

The parser natively produces constituency output. Thus, in order to evaluate the parser on the ICU Finnish corpus as well as to improve its applicability in the domain, we produce a conversion from this constituency scheme to the SD scheme. Note that, as illustrated in Figure 8, using a constituency scheme for ICU Finnish often results in complex representations which do not contain information about syntactic roles of the constituents. Inferring these roles is one of the aims of our conversion.

The conversion is implemented using hand-written rules. The parser assigns a head word for each phrase, and these heads are then used to produce the structure of the dependency graph by placing dependencies from the head word of each constituent to the head words of its sub-constituents. The conversion rules are generally only needed to assign types to these dependencies. There are few exceptions, such as the *sdep* dependencies (see Section 3.2.3) and certain auxiliary structures, where the structure in the SD scheme does not correspond to that induced from the head words. The rules can restrict on the structure of a subtree, that is, a rule can require a phrase as well as its sub-phrases, at any depth, to be of specific types. Our conversion approach closely fol-

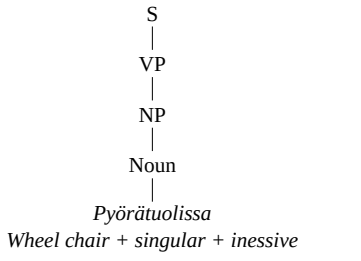

Figure 8: The constituency output of the parser of Laippala et al. (2009). The example sentence can be roughly translated as *In wheel chair*. The direct derivation of the VP from the NP is explained by the missing main verb that would in a corresponding SD analysis be represented by a null verb. Note the size of the tree, despite the fact that the sentence only consists of one word.

lows that of the Stanford tools (de Marneffe et al., 2006), as both utilize heads of phrases and subtree search to produce the structure and labels of the dependency parse.

The conversion rules were developed using the 80-sentence development set previously used by Laippala et al. (2009). We have annotated these sentences in the SD scheme to complement their existing constituency annotation.

6.1.2 Performance of the parser and conversion rules

When interpreting the results it is crucial to note that the rule-based parser does not have a ranking component that would select a single preferred analysis among the generated parses. The parser generates, on average, 33 parses per sentence and the figures reported are measured using the best parse with respect to the labelled attachment score (*oracle performance*). Further, the coverage of the parser in terms of the proportion of sentences that receive at least one analysis is 75% on our corpus and the performance values reported are calculated on these sentences, disregarding sentences that receive no analysis. The results are thus rather an upper limit of the performance to be expected in a real-world setting.

We find that the rule-based parser augmented with our conversion achieves an A_L of 75.2%, A_U of 84.5%, and F-score of 70.2%. Given the A_U of 84.5%, the parser itself assigns incorrect heads for 15.5% of tokens. This is the starting point for the conversion rules, which result in the overall A_L of 75.2%. The difference of 9.3 percentage points between A_U and A_L is divided between errors of the conversion rules and errors of the parser who may assign correct heads but incorrect nonterminal labels, thus preventing correct interpretation of the parse. To establish this division of errors, we have performed a limited manual analysis of 16 randomly selected sentences (75 dependencies) and found that the conversion rules are responsible for 5.3 percentage points and the parser and FinTWOL for the remaining 4 percentage points.

6.2 Statistical parsing experiments with MaltParser

To complement the rule-based dependency parsing experiments, we also apply a statistical parser induced from the ICU Finnish corpus using the MaltParser system[4] (Nivre et al., 2007). We use the arc-eager parsing algorithm characterized as a deterministic, linear-time algorithm that generates a single projective dependency tree in a left-to-right pass through the sentence. The choice of a projective parsing algorithm is justified by the negligible amount of non-projective tokens in the corpus. The algorithm is based on the well-known shift-reduce bottom-up parsing strategy that processes the sentence from a token queue and maintains a stack of partially-processed tokens. At each point in the parsing process, the next transition applied by the parser is decided by a support vector machine (SVM) classifier based on features extracted from the sentence tokens as well as the partially-built dependency tree.

In training the parser, we use the MaltParser default feature model for the arc-eager parsing algorithm. Broadly stated, this model considers morpholexical properties of the first four tokens in the queue and the first two tokens on the stack as well as partially-built dependency structure features of the top items on the stack and the queue. The corpus text is first morphologically disambiguated using FinCG, thus obtaining a single morpholexical reading for each token. A separate feature is then generated for each morpholexical property produced by FinCG[5] for a given token (e.g. the POS, number, and case). Whenever the token wordform does not carry a particular property (e.g. nouns do not have a tense and verbs do not have a case), the

[4] Version 1.2, http://www.maltparser.org

[5] See http://www2.lingsoft.fi/doc/fintwol/intro/tags.html for the full set of tags given by FinTWOL/FinCG

feature is set to *null*. Rather than wordforms, we use word lemmas in the feature model to reduce training data sparseness.

All results reported in this section are obtained using ten-fold cross-validation, where in each fold 80% of the data is used for training, 10% for parameter estimation, and 10% for testing. In preliminary experiments on a small portion of the data, we selected the second degree polynomial kernel for the parser SVM classifier. The values of the SVM regularization parameter C and the kernel parameter γ were selected for each fold separately, using a joint grid search on the parameter estimation set. The best-performing parameter combination in terms of A_L on the parameter estimation set was then used in parsing the test set, thus avoiding parameter over-fitting. All other parameters were left at their default values.

The results are shown in Table 1 for varying sizes of the training sets, in order to estimate the learning curve of the parser. The overall parser performance, 69.9% A_L, can be contrasted with the results of Nivre (2008a) who reports an average A_L of 79.77% across 13 languages. The results for individual languages, however, range from 64.7% for Turkish to 90.1% for Japanese. In that respect, the results for ICU Finnish are among the lower ones, but arguably well within the typical range to be expected. This is particularly encouraging given that the ICU Finnish corpus is currently relatively small, consisting of 1019 sentences and 6082 non-punctuation tokens. As a point of comparison, Nivre has used corpora of 5000 sentences with 58000 tokens, and 17000 sentences with 151000 tokens for Turkish and Japanese, respectively.

The statistical parser yields a lower absolute performance than the rule-based parser. However, the two results are not directly comparable. First, the oracle best-parse strategy had to be used for the rule-based parser. Second, the results of the rule-based parser include only those sentences for which the parser has given at least one analysis (75% of all sentences). Taking these measurement limitations into account, it would seem likely that with a larger corpus available for training and other further improvements, a statistical parsing approach based on MaltParser will be preferable over the rule-based parser of Laippala et al. It is worth noting that the parsing speed of the statistical parser is on the order of 10 sentences per second, whereas the rule-based parser parses one sentence in approximately 2 to 3 seconds.

sample [%]	A_L[%]	A_U [%]	F [%]
100	69.9±2.0	77.1±2.5	66.6±2.2
75	68.4±2.8	75.8±2.2	65.0±3.2
50	65.8±2.0	73.6±1.5	62.0±2.3
25	57.2±2.7	67.5±1.7	52.6±3.2

Table 1: MaltParser results with varying training set size. The *sample* column gives the size to which the training sets in the ten-fold cross-validation were downsampled. Performance figures are given together with their standard deviation on the ten folds.

7 Conclusions and discussion

In this paper, we have presented a new syntactically annotated corpus of ICU Finnish, the language used in daily nursing notes in an intensive care unit. The corpus is annotated in the Stanford dependency scheme which we find suitable for ICU Finnish with only minor modifications. We have performed parsing experiments on this corpus using two approaches: by converting the constituency output of an existing rule-based parser (Laippala et al., 2009) to a dependency scheme, and by inducing a statistical parser from the new corpus using MaltParser (Nivre et al., 2007).

The rule-based parser, together with the constituency-to-dependency conversion developed for the purposes of this work, achieved the oracle labelled attachment score of 75.2%. In a separate evaluation of the conversion rules, we find that the rules contribute roughly 5 percentage points to the overall error rate.

The statistical parser trained on the rather modestly sized corpus achieved a labelled attachment score of 69.9%, approaching the results presented by Nivre (2008a) for parsers trained on significantly larger corpora. The comparability of results of the rule-based and the statistical parsers is difficult to establish given that the rule-based parser does not provide a single preferred analysis.

Our results on the statistical parsing of ICU Finnish, particularly encouraging when taking into consideration the modest size of the corpus, might suggest that full parsing of the intensive care language is, perhaps somewhat counterintuitively, not a very difficult task, relative to the general lan-

guage. For a more definitive conclusion, a considerably broader study, beyond the scope of this paper, would need to be performed. In particular, possible features allowing the parser to better capture the idiosyncrasies of the ICU sublanguage need to be explored more thoroughly.

The first obvious future work direction is to further increase the size of the corpus and find a legal way to release the corpus annotation while protecting patient privacy. One option could, for example, be to release an unlexicalized version of the corpus with morphological and syntactic annotation only. The second direction is to complement the preliminary experiments with MaltParser presented in this paper by carefully exploring the possible feature models, parsing algorithms and parser training parameters in order to maximize the performance of the induced parser. The final direction is to develop a method for inserting the null verbs necessary in the dependency analysis, either as a separate pre-processing step, or directly as part of parsing.

Acknowledgments

We would like to thank Lingsoft Ltd. for making FinTWOL and FinCG available to us. This work has been supported by the Academy of Finland and Tekes, the Finnish Funding Agency for Technology and Innovation.

References

E. Black et al. 1991. A procedure for quantitatively comparing the syntactic coverage of English grammars. In *Proceedings of the Fourth DARPA Speech and Natural Language Workshop*, pages 306–312.

J. E. Carroll, E. Briscoe, and A. Sanfilippo. 1998. Parser evaluation: a survey and a new proposal. In *Proceedings of LREC'98*, pages 447–454.

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

C. Friedman and S. Johnson. 2006. Natural language and text processing in biomedicine. In *Biomedical Informatics*, pages 312–343. Springer.

Y. Huang and H. J. Lowe. 2007. A novel hybrid approach to automated negation detection in clinical radiology reports. *Journal of the American Medical Informatics Association*, 14(3):304–311.

Y. Huang, H. J. Lowe, D. Klein, and R. J. Cucina. 2005. Improved identification of noun phrases in clinical radiology reports using a high-performance statistical natural language parser augmented with the UMLS Specialist Lexicon. *Journal of the American Medical Informatics Association*, 12(3):275–285.

F. Karlsson. 1990. Constraint Grammar as a framework for parsing unrestricted text. In *Proceedings of COLING'90*, pages 168–173.

K. Koskenniemi. 1983. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685.

V. Laippala, F. Ginter, S. Pyysalo, and T. Salakoski. 2009. Towards automatic processing of clinical finnish: A sublanguage analysis and a rule-based parser. *International Journal of Medical Informatics, Special Issue on Mining of Clinical and Biomedical Text and Data*. In press.

D. Lin. 1998. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114.

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

M-C. de Marneffe, B. MacCartney, and C. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *Proceedings of LREC'06*, pages 449–454.

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

J. Nivre. 2008a. Deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553.

J. Nivre. 2008b. Sorting out dependency parsing. In *Proceedings of GoTAL'08*, pages 16–27.

S. Pyysalo, S. Aubin, A. Nazarenko, and T. Salakoski. 2006. Lexical adaptation of Link Grammar to the biomedical sublanguage: a comparative evaluation of three approaches. In *Proceedings of SMBM'06*, pages 60–67.

G. Schneider, F. Rinaldi, and J. Dowdall. 2004. Fast, deep-linguistic statistical dependency parsing. In *Proceedings of COLING'04 Workshop on Recent Advances in Dependency Grammar*, pages 33–40.

P. Szolovits. 2003. Adding a medical lexicon to an English parser. In *Proceedings of AMIA'03*, pages 639–643.

ISSN 1736-6305 Vol. 4
http://hdl.handle.net/10062/9206

Paper II

Dependency-based PropBanking of clinical Finnish.

Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., and Salakoski, T. (2010). In *Proceedings of The Fourth Linguistic Annotation Workshop (LAW IV)*, pages 137–141.

Dependency-based PropBanking of clinical Finnish

Katri Haverinen,[1,3] **Filip Ginter,**[1] **Timo Viljanen,**[1]
Veronika Laippala[2] **and Tapio Salakoski**[1,3]
[1]Department of Information Technology
[2]Department of French Studies
[3]Turku Centre for Computer Science, TUCS
20014 University of Turku, Finland
first.last@utu.fi

Abstract

In this paper, we present a PropBank of clinical Finnish, an annotated corpus of verbal propositions and arguments. The clinical PropBank is created on top of a previously existing dependency treebank annotated in the Stanford Dependency (SD) scheme and covers 90% of all verb occurrences in the treebank.

We establish that the PropBank scheme is applicable to clinical Finnish as well as compatible with the SD scheme, with an overwhelming proportion of arguments being governed by the verb. This allows argument candidates to be restricted to direct verb dependents, substantially simplifying the PropBank construction.

The clinical Finnish PropBank is freely available at the address http://bionlp.utu.fi.

1 Introduction

Natural language processing (NLP) in the clinical domain has received substantial interest, with applications in decision support, patient managing and profiling, mining trends, and others (see the extensive review by Friedman and Johnson (2006)). While some of these applications, such as document retrieval and trend mining, can rely solely on word-frequency-based methods, others, such as information extraction and summarization require a detailed linguistic analysis capturing some of the sentence semantics. Among the most important steps in this direction is an analysis of verbs and their argument structures.

In this work, we focus on the Finnish language in the clinical domain, analyzing its verbs and their argument structures using the PropBank scheme (Palmer et al., 2005). The choice of this particular scheme is motivated by its practical, application-oriented nature. We build the clinical Finnish PropBank on top of the existing dependency treebank of Haverinen et al. (2009).

The primary outcome of this study is the PropBank of clinical Finnish itself, consisting of the analyses for 157 verbs with 2,382 occurrences and 4,763 arguments, and covering 90% of all verb occurrences in the underlying treebank. This PropBank, together with the treebank, is an important resource for the further development of clinical NLP applications for the Finnish language.

We also establish the applicability of the PropBank scheme to the clinical sublanguage with its many atypical characteristics, and finally, we find that the PropBank scheme is compatible with the Stanford Dependency scheme of de Marneffe and Manning (2008a; 2008b) in which the underlying treebank is annotated.

2 The PropBank scheme

Our annotation work is based on the PropBank semantic annotation scheme of Palmer et al. (2005). For each verb, PropBank defines a number of *framesets*, each frameset corresponding to a coarse-grained sense. A frameset consists of a *roleset* which defines a set of *roles* (*arguments* numbered from Arg0 onwards) and their descriptions, and a set of syntactic *frames*. Any element that occurs together with a given verb sufficiently frequently is taken to be its argument. Arg0 is generally a *prototypical Agent* argument and Arg1 is a *prototypical Patient or Theme* argument. The remaining numbered arguments have no consistent overall meanings: they are defined on a verb-by-verb basis. An illustration of a verb with two framesets is given in Figure 1. In addition to the numbered arguments, a verb occurrence can have a number of modifiers, labeled ArgM, each modifier being categorized as one of 14 subtypes, such as *temporal*, *cause* and *location*.

Proceedings of the Fourth Linguistic Annotation Workshop, ACL 2010, pages 137–141,
Uppsala, Sweden, 15-16 July 2010.

kestää.0: “tolerate”	kestää.1: “last”
Arg0: the one who tolerates	Arg1: the thing that lasts
Arg1: what is being tolerated	Arg2: how long it lasts

Figure 1: The PropBank framesets for *kestää* (translated to English from the original frames file) correspond to two different uses of the verb.

Pitkä yövuoro	Long nightshift
Jouduttu laittamaan	Had to put to
illala bipap:lle,	bipap in the evning,
nyt hapettuu hyvin.	now oxidizes well.
DIUREESI: riittävää	DIURESIS: sufficient
Tajunta: rauhallinen	Consciousness: calm
hrhoja ei enää ole	there are no more hllucinations

Figure 2: Example of clinical Finnish (left column) and its exact translation (right column), with typical features such as spelling errors preserved.

3 Clinical Finnish and the clinical Finnish treebank

This study is based on the clinical Finnish treebank of Haverinen et al. (2009), which consists of 2,081 sentences with 15,335 tokens and 13,457 dependencies. The text of the treebank comprises eight complete patient reports from an intensive care unit in a Finnish hospital. An intensive care patient report describes the condition of the patient and its development in time. The *clinical Finnish* in these reports has many characteristics typical of clinical languages, including frequent misspellings, abbreviations, domain terms, telegraphic style and non-standard syntactic structures (see Figure 2 for an illustration). For a detailed analysis, we refer the reader to the studies by Laippala et al. (2009) and Haverinen et al. (2009).

The treebank of Haverinen et al. is annotated in the Stanford Dependency (SD) scheme of de Marneffe and Manning (2008a; 2008b). This scheme is layered, and the annotation variant of the treebank of Haverinen et. al is the *basic* variant of the scheme, in which the analysis forms a tree.

The SD scheme also defines a *collapsed dependencies with propagation of conjunct dependencies* variant (referred to as the *extended* variant of the SD scheme throughout this paper). It adds on top of the *basic* variant a second layer of dependencies which are not part of the strict, syntactic tree. In particular, the *xsubj* dependency marks external subjects, and dependencies involving the heads of coordinations are explicitly duplicated also for the remaining coordinated elements where appropriate. The *extended* variant of the SD scheme is illustrated in Figure 3.

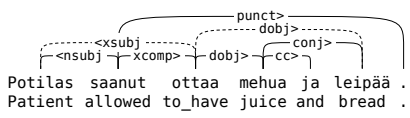

Figure 3: The *extended* SD scheme. The dashed dependencies denote the external subjects and propagated conjunct dependencies that are only part of the *extended* variant of the scheme. The example can be translated as *Patient [has been] allowed to have juice and bread.*

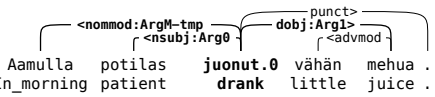

Figure 4: The PropBank annotation scheme on top of the treebank syntactic annotation. The verb *juonut (drank)* is marked with its frameset, in this case the frameset number 0. This frameset specifies that Arg0 marks the agent doing the drinking and Arg1 the liquid being consumed. The ArgM-tmp label specifies that *Aamulla* is a temporal modifier. The example can be translated as *In the morning patient drank a little juice*.

Due to the importance of the additional dependencies for PropBanking (see Section 5 for discussion), we augment the annotation of the underlying treebank to conform to the *extended* variant of the SD scheme by manual annotation, adding a total of 520 dependencies.

The PropBank was originally developed on top of the constituency scheme of the Penn Treebank and requires arguments to correspond to constituents. In a dependency scheme, where there is no explicit notion of constituents, we associate arguments of a verb with dependencies governed by it. The argument can then be understood as the entire subtree headed by the dependent. The annotation is illustrated in Figure 4.

4 PropBanking clinical Finnish

When annotating the clinical Finnish PropBank, we consider all verbs with at least three occurrences in the underlying treebank. In total, we analyze 157 verbs with 192 framesets. Since the treebank does not have gold-standard POS infor-

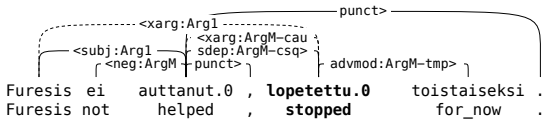

Figure 5: The simplified PropBank annotation strategy. The dashed dependencies labeled with the technical dependency type *xarg* signify arguments and modifiers not in a syntactic relationship to the verb. These arguments and modifiers, as well as those associated with a *conj* or *sdep* dependency (ArgM-csq in this Figure), are only marked in the 100 sentence sample for quantifying unannotated arguments and modifiers. The sentence can be translated as *Furesis did not help, stopped for now*.

mation, we identify all verbs and verbal participles using the FinCG[1] analyzer, which gives a verbal reading to 2,816 tokens. With POS tagging errors taken into account, we estimate the treebank to contain 2,655 occurrences of verbs and verb participles. Of these, 2,382 (90%) correspond to verbs with at least three occurrences and are thus annotated. In total, these verbs have 4,763 arguments and modifiers.

Due to the telegraphic nature of clinical Finnish, omissions of different sentence elements, even main verbs, are very frequent. In order to be able to analyze the syntax of sentences with a missing main verb, Haverinen et al. have added a so called *null verb* to these sentences in the treebank. For instance, the clinical Finnish sentence *Putkesta nestettä (Liquid from the drain)* lacks a main verb, and the insertion of one produces *Putkesta *null* nestettä*. In total, there are 428 null verb occurrences, making the null verb the most common verb in the treebank.

In the clinical PropBank annotation, we treat the null verb in principle as if it was a regular verb, and give it framesets accordingly. For each null verb occurrence, we have determined which regular verb frameset it stands for, and found that, somewhat surprisingly, there were only four common coarse senses of the null verb, roughly corresponding to four framesets of the verbs *olla (to be)*, *tulla (to come)*, *tehdä (to do)* and *laittaa (to put)*. The 26 (6%) null verb occurrences that did not correspond to any of these four framesets were assigned to a "leftover frameset", for which no arguments were marked.

[1]http://www.lingsoft.fi

5 Annotating the arguments on top of the SD scheme

In contrast to the original PropBank, where any syntactic constituent could be marked as an argument, we require arguments to be directly dependent on the verb in the SD scheme (for an illustration, see Figure 5). This restriction is to considerably simplify the annotation process — instead of all possible subtrees, the annotator only needs to look for direct dependents of the verb. In addition, this constraint should naturally also simplify possible automatic identification and classification of the arguments.

In addition to restricting arguments to direct dependents of the verb, coordination dependencies *conj* and *sdep* (implicit coordination of top level independent clauses, see Figure 5) are left outside the annotation scope. This is due to the nature of the clinical language, which places on these dependencies cause-consequence relationships that require strong inference. For instance, sentences such as *Patient restless, given tranquilizers* where there is clearly a causal relationship but no explicit marker such as *thus* or *because*, are common.

Naturally, it is necessary to estimate the effect of these restrictions, which can be justified only if the number of lost arguments is minimal. We have conducted a small-scale experiment on 100 randomly selected sentences with at least one verb that has a frameset assigned. We have provided this portion of the clinical PropBank with a full annotation, including the arguments not governed by the verb and those associated with *conj* and *sdep* dependencies. For an illustration, see Figure 5.

There are in total 326 arguments and modifiers (169 arguments and 157 modifiers) in the 100 sentence sample. Of these, 278 (85%) are governed by the verb in the *basic* SD scheme and are thus in a direct syntactic relationship with the verb. Fur-

ther 19 (6%) arguments and modifiers are governed by the verb in the *extended* SD scheme. Out of the remaining 29 (9%), 23 are in fact modifiers, leaving only 6 numbered arguments not accounted for in the *extended* SD scheme. Thus, 96% (163/169) of arguments and 85% (134/157) of modifiers are directly governed by the verb.

Of the 23 ungoverned modifiers, all are either cause (CAU) or consequence (CSQ)[2]. Of the *sdep* and *conj* dependencies only a small portion (9/68) were associated with an argument or a modifier, all of which were in fact CAU or CSQ modifiers. Both these and the CAU and CSQ modifiers not governed by the verb reflect strongly inferred relationships between clauses.

Based on these figures, we conclude that an overwhelming majority of arguments and modifiers is governed by the verb in the *extended* SD scheme and restricting the annotation to dependents of the verb as well as leaving *sdep* and *conj* outside the annotation scope seems justified. Additionally, we demonstrate the utility of the conjunct dependency propagation and external subject marking in the *extended* SD scheme.

6 Related work

Many efforts have been made to capture meanings and arguments of verbs. For instance, the VerbNet project (Kipper et al., 2000) strives to create a broad on-line verb lexicon, and FrameNet (Ruppenhofer et al., 2005) aims to document the range of valences of each verb in each of its senses. The PropBank project (Palmer et al., 2005) strives for a practical approach to semantic representation, adding a layer of semantic role labels to the Penn Treebank (Marcus et al., 1993).

In addition to the original PropBank by Palmer et al., numerous PropBanks have been developed for languages other than English (e.g. Chinese (Xue and Palmer, 2003) and Arabic (Diab et al., 2008)). Also applications attempting to automatically recover PropBank-style arguments have been proposed. For example, the CoNLL shared task has focused on semantic role labeling four times, twice as a separate task (Carreras and Màrquez, 2004; Carreras and Màrquez, 2005), and twice in conjunction with syntactic parsing (Surdeanu et al., 2008; Hajič et al., 2009).

[2]CSQ is a new modifier subtype added by us, due to the restriction of only annotating direct syntactic dependents, which does not allow the annotation of all causal relationships with the type CAU.

In semantic analysis of clinical language, Paek et al. (2006) have experimented on PropBank-based machine learning on abstracts of Randomized Controlled Trials (RCTs), and Savova et al. (2009) have presented work on temporal relation discovery from clinical narratives.

7 Conclusion

In this paper, we have presented a PropBank of clinical Finnish, building a new layer of annotation on top of the existing clinical treebank of Haverinen et al. (2009). This PropBank covers all 157 verbs occurring at least three times in the treebank and accounts for 90% of all verb occurrences.

This work has also served as a test case for the PropBank annotation scheme in two senses. First, the scheme has been tested on a highly specialized language, clinical Finnish, and second, its compatibility with the SD syntactic scheme has been examined. On both accounts, we find the PropBank scheme a suitable choice.

In general, the specialized language did not seem to cause problems for the scheme. For instance, the frequent null verbs could be analyzed similarly to regular verbs, with full 94% belonging to one of only four framesets. This is likely due to the very restricted clinical domain of the corpus.

We also find a strong correspondence between the PropBank arguments and the verb dependents in the *extended* SD scheme, with 96% of arguments and 85% of modifiers being directly governed by the verb. The 15% ungoverned modifiers are cause-consequence relationships that require strong inference. This correspondence allowed us to simplify the annotation task by only considering direct verb dependents as argument candidates.

The new version of the treebank, manually anonymized, including the enhanced SD scheme annotation and the PropBank annotation, is freely available at `http://bionlp.utu.fi`.

Acknowledgments

We are grateful to Heljä Lundgren-Laine, Riitta Danielsson-Ojala and prof. Sanna Salanterä for their assistance in the anonymization of the corpus. We would also like to thank Lingsoft Ltd. for making FinTWOL and FinCG available to us. This work was supported by the Academy of Finland.

References

Xavier Carreras and Lluís Màrquez. 2004. Introduction to the CoNLL-2004 shared task: Semantic role labeling. In *HLT-NAACL 2004 Workshop: Eighth Conference on Computational Natural Language Learning (CoNLL-2004)*, pages 89–97, Boston, Massachusetts, USA, May 6 - May 7. Association for Computational Linguistics.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the CoNLL-2005 shared task: Semantic role labeling. In *Proceedings of the Ninth Conference on Computational Natural Language Learning (CoNLL-2005)*, pages 152–164, Ann Arbor, Michigan, June. Association for Computational Linguistics.

Marie-Catherine de Marneffe and Christopher Manning. 2008a. Stanford typed dependencies manual. Technical report, Stanford University, September.

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

Mona Diab, Mansouri Aous, Martha Palmer, Babko-Malaya Olga, Wadji Zaghouani, Ann Bies, and Mohammed Maamouri. 2008. A pilot Arabic PropBank. In *Proceedings of LREC'08*, pages 3467–3472. Association for Computational Linguistics.

Carol Friedman and Stephen Johnson. 2006. Natural language and text processing in biomedicine. In *Biomedical Informatics*, pages 312–343. Springer.

Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria A. Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Niawen Xue, and Yi Zhang. 2009. The CoNLL-2008 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09: Shared Task*, pages 1–18. Association for Computational Linguistics.

Katri Haverinen, Filip Ginter, Veronika Laippala, and Tapio Salakoski. 2009. Parsing clinical Finnish: Experiments with rule-based and statistical dependency parsers. In *Proceedings of NODALIDA'09, Odense, Denmark*, pages 65–72.

Karin Kipper, Hoa Trang Dang, and Martha Palmer. 2000. Class-based construction of a verb lexicon. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence*, pages 691–696. AAAI Press / The MIT Press.

Veronika Laippala, Filip Ginter, Sampo Pyysalo, and Tapio Salakoski. 2009. Towards automatic processing of clinical Finnish: A sublanguage analysis and a rule-based parser. *International Journal of Medical Informatics, Special Issue on Mining of Clinical and Biomedical Text and Data*, 78:7–12.

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

Hyung Paek, Yacov Kogan, Prem Thomas, Seymor Codish, and Michael Krauthammer. 2006. Shallow semantic parsing of randomized controlled trial reports. In *Proceedings of AMIA'06*, pages 604–608.

Martha Palmer, Dan Gildea, and Paul Kingsbury. 2005. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1).

Josef Ruppenhofer, Michael Ellsworth, Miriam R. L. Petruck, Christopher R. Johnson, and Jan Scheffczyk. 2005. FrameNet II: Extended theory and practice. Technical report, ICSI.

Guergana Savova, Steven Bethard, Will Styler, James Martin, Martha Palmer, James Masanz, and Wayne Ward. 2009. Towards temporal relation discovety from the clinical narrative. In *Proceedings of AMIA'09*, pages 568–572.

Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. 2008. The CoNLL-2008 shared task on joint parsing on syntactic and semantic dependencies. In *Proceedings of CoNLL'08*, pages 159–177. Association for Computational Linguistics.

Nianwen Xue and Martha Palmer. 2003. Annotating the propositions in the Penn Chinese Treebank. In *Proceedings of the 2nd SIGHAN Workshop on Chinese Language Processing*, pages 47–54, Sapporo, Japan. Association for Computational Linguistics.

Paper III

Dependency annotation of Wikipedia: First steps towards a Finnish treebank.

Haverinen, K., Ginter, F., Laippala, V., Viljanen, T., and Salakoski, T. (2009). In *Proceedings of The Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 95–105.

Dependency Annotation of Wikipedia: First Steps Towards a Finnish Treebank

Katri Haverinen,[1,3] Filip Ginter,[1] Veronika Laippala,[2]
Timo Viljanen,[1] Tapio Salakoski[1,3]

[1]Department of Information Technology,
[2]Department of French studies
[3]Turku Centre for Computer Science (TUCS)
20014 University of Turku, Finland
first.last@utu.fi

Abstract

In this work, we present the first results obtained during the annotation of a general Finnish treebank in the Stanford Dependency scheme. We find that the scheme is a suitable syntax representation for Finnish, with only minor modifications needed. The treebank is based on text from the Finnish Wikipedia, ensuring its free distribution and broad topical variance. To assess the suitability of Wikipedia text as the basis of a treebank, we analyze its grammaticality and find the quality of the language surprisingly high, with 97.2% of the sentences judged as grammatical. The treebank currently consists of 60 fully annotated articles and is freely available.

1 Introduction

Treebanks are among the most crucial resources for the development of natural language processing (NLP) methods. There exist a number of national treebanks for a variety of languages, including widely used and studied ones, such as English, as well as languages spoken by comparatively smaller populations, for example Slovene. For Finnish, no such treebank currently exists, considerably restricting the possibilities for NLP research for this language. To address this obvious deficiency, we have commenced an effort to develop the first Finnish language treebank and, in this paper, present the first results of this project.

The source of text for the treebank is the Finnish Wikipedia. One of its major advantages is that it is released under a free license, enabling the distribution of the resulting treebank at no cost and with no copyright issues. Apart from offering a great topical variety, the text is written collaboratively by a number of authors and thus also reflects a number of different personal writing styles. Since there is little

prior work on Wikipedia-based treebanking, we assess the grammaticality of the language and thus, to some extent, its suitability for a source of treebank text.

The annotation scheme of the treebank is the well-known Stanford Dependency (SD) scheme which was designed specifically for NLP applications [1, 10]. The Finnish treebank is the first general language corpus annotated natively in the SD scheme. Since the scheme was originally designed for English, we discuss its applicability to Finnish as part of the results presented in this paper. In particular, we show that only minor modifications to the scheme are necessary. The choice of the scheme follows a recent substantial interest in the application of dependency schemes in general and the numerous successful applications of the SD scheme specifically [8, 10, 12].

Among the most important application areas for treebanks is the induction and evaluation of statistical parsers. For instance, a number of national treebanks for diverse languages such as Catalan, English, and Japanese have been used in the recent CoNLL'09 shared task [2] to develop and evaluate multilingual statistical parsers, thus greatly benefiting the NLP research for these languages. Indeed, one of the primary motivations for this work is to provide a similar opportunity for Finnish NLP research. This motivation has affected both the choice of the scheme and the target size of the corpus, as will be discussed later.

2 Related work

As stated earlier, there is no publicly available treebank of general Finnish. The only treebank we are aware of is that of Haverinen et al. [4] who have applied the SD scheme to Finnish intensive care nursing narratives, producing a treebank of 1019 sentences. This treebank, however, is not publicly available due to patient privacy issues.

Also other NLP resources for Finnish are scarce. The only broad-coverage full syntactic parser for Finnish is the closed source commercial parser Connexor Syntax.[1] Other NLP tools, particularly targeted at morphological analysis, include FinTWOL and FinCG,[2] a morphological analyzer and a Constraint Grammar parser which resolves morphological ambiguity [5, 6], both commercial products. In addition, a rule-based parser has been developed by Laippala et al. [7], particularly targeting the language used in nursing narratives in a Finnish intensive care unit. This parser is, however, restricted to the very specific vocabulary and syntax typical for this domain.

Apart from the nursing narrative corpus of Haverinen et al., there is a second treebank with SD as its native annotation scheme, BioInfer [13]. It is an English-language corpus of 1100 sentences from research article abstracts focusing on protein-protein interactions. In addition to these two corpora, any treebank

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

[2] http://www.lingsoft.fi

annotated in the Penn Treebank [9] scheme can be automatically converted to the SD scheme using the method and tools[3] of de Marneffe and Manning [10].

3 Adaptation of the SD scheme to Finnish

In this section, we introduce our modifications to the Stanford Dependency scheme. Sections 3.2 and 3.3 discuss Finnish-specific adjustments, while Sections 3.4 through 3.7 consider more general modifications. Due to space limitations, the original SD scheme will only be discussed briefly, and the reader is referred to the work of de Marneffe and Manning [1] for a thorough description.

3.1 The Stanford Dependency scheme

In the SD scheme, the syntactic structure of a sentence is represented as a directed graph of labelled dependencies. The latest scheme version [1] defines 55 hierarchically arranged dependency types, capturing both syntactic and semantic relations. There are four different representation variants, in which different sets of dependencies are present. In the basic variant, used in the current annotation, the analyses are trees and generally include only syntactic dependencies. Other variants define a number of additional, semantically motivated dependency types that are present in addition to the basic syntactic dependencies. These variants thus result in non-tree structures that may even contain directed cycles.

The scheme is designed to be application-oriented and has indeed proved its usefulness in a number of NLP methods (for an extensive list, see the review by de Marneffe and Manning [10]). These successful applications have also contributed to our decision of using the scheme in this work, as has the encouraging observation that the SD scheme would seem to be suitable at least for clinical Finnish, as reported by Haverinen et al. [4].

Haverinen et al. adapted the SD scheme to clinical Finnish by introducing several new dependency types that address the most common Finnish syntactic structures that the SD scheme could not naturally represent: inflected nominal modifiers, adpositional phrases, and certain passive structures (for details, see [4]). These modifications apply with no further changes also to general Finnish, and, in the following, we discuss our additional adaptations of the scheme.

3.2 Genitive objects

In Finnish, a noun with a verb counterpart or a nominalization of a verb can have an object, called the *genitive object*. This resembles the English phenomenon where a gerundial noun takes an object in front of it, as in *ship building*, except that the genitive case is not used in the English structure. In English, nominal pre-modifiers

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

such as the above are considered syntactic compounds and are marked *nn* in the SD scheme.

On the surface, genitive objects are identical to possessive modifiers, both being nominal pre-modifiers in the genitive case. There is, however, a clear semantic difference between these two. For instance, the possessive interpretation of *laivan rakentaminen* (*ship+genitive building*) would mean that the ship itself is doing the building, whereas the genitive object interpretation would mean that the ship is being built. In order to maintain this semantic distinction, it is necessary to establish a new dependency type, *gobj*, for genitive objects.

3.3 Finnish copulas

The SD scheme reserves a special treatment for copula structures: the predicative of a copular clause is the head and the copular verb its dependent. In all other cases, the finite verb acts as the head. This is motivated from a multilingual point of view, as not all languages have an overt copular verb. Further, particularly in telegraphic style, the copular verb can often be omitted even in those languages that do. This treatment of copula structures, however, requires an exact definition of the class of copular verbs and predicatives.

The SD scheme uses a list of English copular verbs defined in the Penn Treebank, including, among others, *to be*, *to resemble* and *to become*. According to Finnish Grammar [3, §891], the only Finnish copular verb is *olla* (*to be*), and all clauses with *olla* as the main verb can be classified as copular. This includes clauses where the predicative is inflected in a local case, such as *Paketti on Oulusta* (*The_package is from_Oulu*). However, if a structure such as this one is accepted as copular, a sentence with several possible predicatives, such as *Paketti on Oulusta ystävältäni* (*The_package is from_Oulu from_my_friend*) can easily be formed. Such a structure has no obvious dependency representation in the SD scheme, since the clause would have two head words. Another problem related to the predicative cases is that of distinguishing the copular verb *olla* (*to be*) and other, non-copular verbs that take as their argument a noun inflected in the same case as the argument of the verb *olla*. Consider, for example, *olla laulajana* (*to_be singer+essive*), *toimia laulajana* (*to_act as_singer+essive*) and *työskennellä laulajana* (*to_work as_singer+essive*). All three examples have the same surface syntactic structure, yet for instance the third example is certainly not a case of copula.

To avoid the class of copulas becoming unnecessarily broad, and syntactically and semantically diverse, we only allow nominative and partitive cases for noun and adjective predicatives, which permits us to restrict copular structures to those that include the only Finnish copular verb, *olla*. In addition to nouns and adjectives, for instance adverbs and even full clauses can act as predicatives. Our solution, including our use of the separate copula subject type, *nsubj-cop*, is similar to that in the clinical treebank of Haverinen et al., although some of the most problematic cases do not occur in the clinical language. For an illustration of our analysis of Finnish copula structures, see Figure 1.

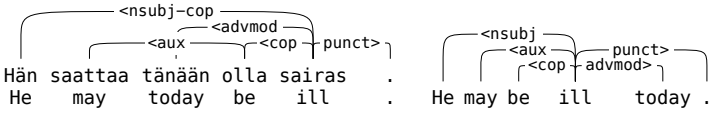

Figure 1: Finnish copula structures (left) as compared to those of English (right). Note that the copula acts as the head for the possible auxiliary which can sometimes cause non-projective structures. Also note the use of the *nsubj-cop* dependency type.

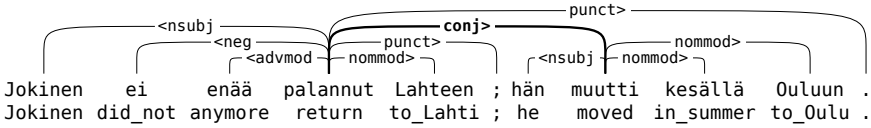

Figure 2: Implicit clausal coordination. The example sentence could be translated as "Jokinen did not return to Lahti anymore; he moved to Oulu in the summer."

3.4 Independent clause coordination

Independent clauses can be coordinated without a conjunction, as in *Lapset pyöräilivät kouluun; aikuiset ajoivat töihin* (*The children cycled to school; the adults drove to work*). The SD scheme analyzes such implicit coordination as parataxis and defines the corresponding dependency type. We, however find these structures functionally and semantically similar to explicit coordinations and thus also annotate them similarly. This is particularly natural in the SD scheme which analyzes conjunctions as mere dependents of the first coordinated element, making implicit and explicit coordinations differ only in the presence or absence of this single dependent. The *parataxis* type is then reserved for other types of parataxis such as reporting clauses. In this respect the scheme also diverts from that used by Haverinen et al., who defined a separate dependency type, *sdep*, for implicit clause coordination. Our analysis is illustrated in Figure 2.

3.5 Infinite clausal complements

The original SD scheme does not distinguish between finite and infinite clausal complements, but uses the type *ccomp* for both. For instance, in the structures *Sanoin, että pallo katosi* (*I_said that the_ball disappeared*) and *Estin palloa katoamasta* (*I_prevented the_ball from_disappearing*), the complements *että pallo katosi* (*that the_ball disappeared*) and *palloa katoamasta* (*the_ball from_disappearing*) would both be analyzed as *ccomp* in the original SD scheme. The *iccomp* dependency type enables the distinction of these two structures, which would otherwise not be possible without morphological information that, currently, is not present in the treebank.

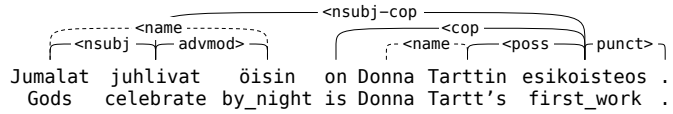

Figure 3: *Jumalat juhlivat öisin* (*Gods celebrate by night*) is a named entity with an inner syntactic structure and is thus given a full syntactic analysis, including the correct head word. *Donna Tarttin* is only marked as a multi-word unit with no further analysis. The technical dependency *name* is used to delimit named entity boundaries.

3.6 Named entities

Multi-word named entities, such as names of people, cities, books, and movies, are frequent in general language. These elements are problematic in a number of ways: often, but not always, they lack an obvious inner syntactic structure, despite consisting of several words, as for example *Carl Gustaf Emil Mannerheim*, or they may also be in another language, like *Västra Finnholmen*. An example of a name that does have a complex inner structure and is in Finnish would be the name of the book *Taistelu sosiaaliturvasta — ammattiyhdistysväen toiminta sosiaaliturvan puolesta 1957–1963* (in English *The battle for social security — trade union members' actions for social security 1957–1963*).

All multi-word names are annotated as single units whose rightmost word acts as the head in the dependency tree. In addition, Finnish names that do have an inner syntactic structure are given a full dependency annotation and their correct head word is identified (see Figure 3 for an illustration). This approach thus leaves open two options for treating Finnish named entities with inner structure. One possibility is to discard the annotation of the inner structure and consequently treat the named entities as single units. The other alternative is to preserve these entities as subtrees in the syntactic structure. The choice will likely be application-dependent.

3.7 Gapping and fragments

Gapping, a form of ellipsis where a governing element is omitted to avoid repetition while its dependents are not, poses an annotation problem. For instance, in *minä söin jäätelöä ja sinä salaattia* (*I ate ice cream and you salad*), the elided verb is necessary to construct a tree that correctly reflects the meaning of the sentence. A similar case is that of fragments, such as *Presidentti Kiinaan* (*The President to China*), where the head word of the clause is absent.

In order to be able to construct an analysis for such cases, we insert a *null* token into the sentences to represent the missing head word. In the case of gapping, where the antecedent of the elided element is present earlier in the sentence, we further include a semantic dependency, *ellipsis*, to relate the antecedent and the *null* token. In the case of fragments, no antecedent is present in the sentence and consequently

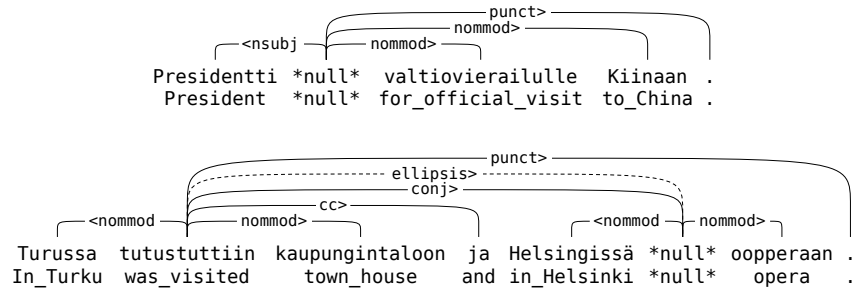

Figure 4: Null tokens in the case of fragments (top) and gapping (bottom). Note the semantic dependency *ellipsis*. The fragment sentence could be translated as "The President for official visit to China" and the ellipsis sentence as "The town house was visited in Turku and the opera in Helsinki."

the *ellipsis* dependency is not used. Figure 4 illustrates the usage of *null* tokens.

Note that the *null* tokens are only used to stand for missing governors. Consequently, other elements that do not generally act as governors in the SD scheme, such as missing copula verbs and auxiliaries, are not represented by *null* tokens, neither are other forms of ellipsis.

4 Construction of the Treebank

In this section, we describe the ongoing work on the Finnish dependency treebank itself.

4.1 Treebank text

In constructing the treebank, we use randomly selected articles from the Finnish Wikipedia. All articles that do not exceed 75 sentences are annotated in their entirety, excluding parts that do not have enough syntactic structure to annotate, such as bulleted lists of single words, section headings and figure captions. Longer articles are truncated at 75 sentences, to keep the treebank from becoming biased towards a single article topic.

Currently, we have completed the annotation of 60 articles, comprising 711 sentences and 10217 tokens, of which 8801 are non-punctuation. Thus, on average, one article is 11 sentences long and one sentence contains 14 tokens. The length of articles varies substantially — the longest article in the currently annotated part is 61 sentences long, while the shortest contains only one sentence. Out of all sentences in the treebank, 27 (3.8%) are non-projective. For comparison, Haverinen et al. report the proportion of non-projective sentences in the clinical treebank to be 2.9%. The currently existing annotation, subject to further changes, is available at `http://bionlp.utu.fi/fintreebank.html` to illustrate

the annotation scheme.

4.2 The Annotation process

In our annotation work, we use a custom annotation tool, which will be made publicly available together with the treebank. It includes the basic abilities necessary for dependency annotation, along with search abilities and the possibility to mark a dependency for later discussion, or label a sentence as dubious or ungrammatical.

We have started the annotation process by first annotating 562 sentences (47 articles) in trial annotations. Each sentence was first annotated by one annotator and the annotation was then jointly inspected by the whole group. Authoritative decisions were made for all problematic cases found at this stage, and the already existing annotation was modified as necessary to ensure its consistency.

After the trial annotations, full double annotation has been started. That is, each sentence is first independently annotated by two annotators and all differences are then jointly resolved. The decisions made at this double annotation stage lean on the authoritative decisions made after the trial annotations. The current number of double annotated sentences is 149 (13 articles). Due to the currently small number of these sentences, we do not report an inter-annotator agreement at this stage, as this figure would not be representative. Inter-annotator agreement in the double annotation will be measured regularly throughout the annotation process to estimate the annotation quality and will be reported with the final release of the corpus.

5 Characteristics of Wikipedia text

The text in Wikipedia articles is sometimes thought to be of poor quality with respect to grammaticality. To determine some properties of the Wikipedia language, we have conducted a small-scale analysis of the currently annotated sample, estimating the proportion of spelling and grammar errors.

We assess the amount of spelling errors in the text by manually inspecting all words that FinTWOL,[4] a broad-coverage morphological analyzer, failed to recognize. Of the 1034 (10.1% of all tokens) unrecognized tokens, only 6 (0.6‰) were obvious misspellings, the remaining being most commonly names, foreign words, numerical expressions, untypical punctuation symbols, abbreviations, etc.

To estimate the level of ungrammaticality in the Wikipedia text, each sentence was assessed independently by three native speaker annotators, and marked *grammatical*, *questionable* or *ungrammatical*. All sentences not judged grammatical by at least two of the three annotators were further manually analyzed to determine the type of error they contained. The results of this manual analysis are given in Table 1. The vast majority of sentences, 691 out of 711 (97.2%), were judged grammatical by at least two annotators; 627 (88.2%) were judged grammatical unanimously. Further, 18 sentences (2.5%) were judged questionable and

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

Mistake type	Frequency
Fragment	6
Relative clause error	4
Compound error	3
Translation error, anglicism or colloquial	6
Inflection error	2
Coordination error	2
Total	23

Table 1: Results of the manual analysis of grammar errors. Note that the total number of errors is greater than the total number of ungrammatical and questionable sentences, as some sentences had more than one error in them.

2 (0.3%) ungrammatical. Out of the 20 sentences not judged grammatical, only one was downright incomprehensible. Fragments are among the most common cases judged questionable or ungrammatical, as are translation errors, anglicisms and colloquial language.

In general, many sentences judged as questionable were colloquial rather than strictly erroneous. Examples of such colloquial structures, which would in some contexts be judged ungrammatical, can be a sizeable asset for example when building a parser targeting text produced by non-professional writers. To conclude, we find the overall quality of the Wikipedia text, in terms of grammaticality and correct spelling, clearly acceptable.

6 Conclusions and future work

In this paper, we have presented first results of an ongoing effort to build a treebank of the Finnish language. First, we demonstrate that the Stanford Dependency scheme is applicable to general Finnish with only minor modifications. Many of these modifications have previously been introduced by Haverinen et al. [4] who applied the SD scheme to Finnish nursing narratives. Second, we assess the grammaticality of the Finnish Wikipedia language and find it, maybe somewhat surprisingly, clearly acceptable. In addition to the obvious benefit that Wikipedia text is freely available under an open license, it may also be an asset for a number of real-world applications that the language found in the articles can be colloquial and is not necessarily produced by professional writers. Currently, the treebank consists of 60 fully annotated articles, comprising of 711 sentences. The annotation is available at `http://bionlp.utu.fi/fintreebank.html`.

The primary goal of the project is to create a freely available treebank large enough for the induction of a broad-coverage statistical parser as well as the development of natural language processing methods in general. The first and most important future work direction is thus naturally to increase the size of the corpus.

Currently, we aim at annotating roughly 10,000 sentences, that is, about 140,000 tokens, a treebank size shown to be sufficient to induce an accurate statistical parser for a number of languages [11]. The performance and learning curve of the induced parser and other NLP methods that use the treebank will help to determine its final size.

A second, more long-term direction is to further enhance the annotation of the treebank by providing a layer of more detailed semantic analysis, for example using an SD scheme variant that also includes semantically oriented dependency types. In this layer, it would also be possible to deepen the annotation of elliptic structures by marking also omission of non-head elements. This will require further modifications to the SD scheme which does not prescribe any treatment of ellipsis. Thirdly, the possibility to provide morphological and POS information for the treebank using an existing analyzer for Finnish will be investigated.

Acknowledgements

We would like to thank Lingsoft Ltd. for making FinTWOL available to us. This work has been supported by the Academy of Finland and Turun Yliopistosäätiö.

References

[1] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies manual. Technical report, Stanford University, September 2008.

[2] Jan Hajič, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Štěpánek, Pavel Straňák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The CoNLL 2009 shared task: Syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09*, pages 1–18, 2009.

[3] Auli Hakulinen, Maria Vilkuna, Riitta Korhonen, Vesa Koivisto, Tarja-Riitta Heinonen, and Irja Alho. *Iso suomen kielioppi / Grammar of Finnish*. Suomalaisen kirjallisuuden seura, 2004.

[4] Katri Haverinen, Filip Ginter, Veronika Laippala, and Tapio Salakoski. Parsing clinical Finnish: Experiments with rule-based and statistical dependency parsers. In *Proceedings of NODALIDA'09, Odense, Denmark*, 2009.

[5] Fred Karlsson. Constraint Grammar as a framework for parsing unrestricted text. In *Proceedings of COLING'90*, pages 168–173, 1990.

[6] Kimmo Koskenniemi. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[7] Veronika Laippala, Filip Ginter, Sampo Pyysalo, and Tapio Salakoski. Towards automatic processing of clinical Finnish: A sublanguage analysis and a rule-based parser. *International Journal of Medical Informatics, Special Issue on Mining of Clinical and Biomedical Text and Data*, 2009. In press, available in online version only.

[8] Dekang Lin. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114, 1998.

[9] Mitchell Marcus, Mary Ann Marcinkiwicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2), 1993.

[10] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies representation. In *Proceedings of COLING'08, Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, 2008.

[11] Joakim Nivre. Deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[12] Joakim Nivre. Sorting out dependency parsing. In *Proceedings of GoTAL'08*, pages 16–27, 2008.

[13] Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP'07*, pages 25–32, 2007.

Paper IV

Treebanking Finnish.

Haverinen, K., Viljanen, T., Laippala, V., Kohonen, S., Ginter, F., and Salakoski, T. (2010). In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*, pages 79–90.

Treebanking Finnish

Katri Haverinen,[1,3] Timo Viljanen,[1] Veronika Laippala,[2]
Samuel Kohonen,[1] Filip Ginter[1] and Tapio Salakoski[1,3]

[1]Department of Information Technology,
[2]Department of French studies
[3]Turku Centre for Computer Science (TUCS)
20014 University of Turku, Finland
first.last@utu.fi

Abstract

In this paper, we present the current version of a syntactically annotated corpus for Finnish, the Turku Dependency Treebank (TDT). This is the first publicly available Finnish treebank of practical size, currently consisting of 4,838 sentences (66,042 tokens). The treebank includes both morphological and syntactic analyses, the morphological information being produced using the FinCG analyzer, and the syntax being human-annotated in the Stanford Dependency scheme. Additionally, we conduct an experiment in automatic pre-annotation and find the overall effect positive. In particular, pre-annotation may be tremendously helpful in terms of both speed and accuracy for an annotator still in training, although for more experienced annotators such obvious benefit was not observed.

In addition to the treebank itself, we have constructed a custom annotation software, as well as a web-based interface with advanced search functions. Both the treebank, including the full edit-history with exact timings, and its associated software are publicly available under an open license at the address http://bionlp.utu.fi.

1 Introduction

The applications of treebanks and their benefits for natural language processing (NLP) are numerous and well-known. Many languages, regardless of how widely spoken, already have a treebank, and for many others one is currently being developed. Finnish is among the less fortunate languages in the sense that it previously long lacked a publicly available treebank entirely. Even now, prior to this work, the only such treebank is our previously published small-scale treebank [3], which does not yet truly enable NLP research.

In this work, we aim to address the serious lack of NLP resources for Finnish, by extending our previous work into a freely available, practically sized treebank

for Finnish, the Turku Dependency Treebank (TDT). The current, extended version of the treebank presented in this paper includes 4,838 sentences. The whole treebank has manually created syntax annotations in the well-known Stanford Dependency (SD) scheme [1, 9] and automatically created morphological analyses. The text of the treebank is drawn from four sources: the Finnish Wikipedia and Wikinews, popular blogs and a university web-magazine.

As a second contribution, we also conduct an experiment on the effect of automated pre-annotation on annotation speed and quality, by using a preliminary statistical parser induced from the treebank to produce an initial analysis.

The linguistic aspects of the work, such as the choice of the annotation scheme and the modifications needed to accommodate the specific features of the Finnish language have been thoroughly discussed in our previous paper on the first release of the treebank [3]. In particular, we have found the Stanford Dependency scheme suitable for the Finnish language, with only minor modifications needed. Thus this paper will rather focus on the annotation process point of view of the work.

2 Related Work

The only publicly available treebank of general Finnish is our previously released treebank version [3]. This version only consists of 711 sentences, and, unlike the extended treebank release presented here, lacks morphological information. Also, no inter-annotator agreement figures were presented for this previous release. In addition to the general Finnish treebank, there exists a recently published small-scale treebank and PropBank of clinical Finnish [4]. The size of this corpus is 2,081 sentences (15,335 tokens), and it includes morphological, syntactic and semantic annotation.

Due to the lack of a large, publicly available treebank, also Finnish NLP tools are scarce. Tools targeted at Finnish morphology include FinTWOL and FinCG, a commercial morphological analyzer and a constraint grammar parser that resolves morphological ambiguity [5, 6]. These tools are used in this work to provide morphological analyses for the treebank. The only previously available broad-coverage syntactic parser for Finnish is Machinese Syntax,[1] which is a closed-source commercial parser.

The syntactic representation scheme used in this work, the Stanford Dependency (SD) scheme [1, 9], is relatively widely used in NLP applications. Both the above mentioned treebanks of Finnish use this scheme and additionally, there is a third treebank that has native SD annotation. The BioInfer [13] treebank is an English language corpus of scientific abstracts in the biomedical domain. In addition to these native corpora, also any English language treebank that uses the Penn Treebank scheme [8] can be automatically converted into the SD scheme using existing tools[2].

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

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

3 Treebank Text

In the current version of the treebank, there are four distinct sources of text: the Finnish Wikipedia and Wikinews, popular blogs and a university web-magazine. These sources are selected on the basis of two criteria.

First, it is our fundamental belief that the treebank should be freely available under an open license, which restricts our choice of texts to those which either are published under an open license originally or for which we can, with reasonable effort, negotiate such a license. Three of the current sources, Wikipedia, Wikinews and the university web-magazine, were originally published under an open license, and for the blog texts, we have obtained the permission to re-publish the text from individual authors.

Second, we have strived for linguistic variety in the texts. We have specifically limited the amount of text chosen about the same topic, and by the same author. In Wikipedia, this is naturally achieved by choosing the articles randomly, as both the amount of articles and the amount of authors are large. In the Finnish Wikinews, the number of authors is substantially smaller, and thus we have, when choosing an article from this source, first randomly selected an author, and only after that randomly selected one individual article by them. This selection process was repeated until a sufficient amount of articles had been chosen.

When selecting the blog texts, we have used several lists of most popular blogs and only selected blogs where the entries appeared to be of sufficient grammatical quality to allow proper annotation of syntax. In addition, the blogs were divided into categories, based on which topic the majority of the entries were about, and the amount of blogs selected from each category was limited. The current selection consists of two blogs from the category *personal and general*, one from the category *style and fashion* and one from the category *relationships and sex*. Naturally, this selection was affected by the permissions given by the authors. The individual texts were selected starting from the newest entries, discarding entries containing certain problematic properties, such as long quotes which could cause copyright issues. We limited the amount of text to be selected from one blog author to be approximately 200 sentences, so that individual entries from a blog were selected in order until the total amount of sentences surpassed 200. Thus the amount of entries chosen from each blog varies according to the length of the entries in that blog.

In the case of the university web magazine, articles were selected starting from the newest writings. Given the relatively limited topics, this section was restricted to a total of 50 articles, which results in a total of 942 sentences.

The breakdown of articles and sentences in different sections of the treebank is shown in Table 1. The table shows that the largest section of the treebank is currently the Wikipedia section, followed by the Wikinews section and the university web-magazine section. The smallest section is at the moment the blog section, which is due to the difficulty of gaining re-publication permissions from individual authors. Altogether the current version of the treebank consists of 4,838 sentences

Section	articles	sentences	tokens
Wikipedia	199	2,260	32,111
Wikinews	67	760	9,724
Blogs	32	876	10,918
Web-magazine	50	942	13,289
Total	348	4,838	66,042

Table 1: Breakdown of the treebank sections. As the annotation work still continues, it should be noted that the current breakdown of sections does not reflect the final composition of the treebank.

(66,042 tokens). Out of all sentences, 5.8% are non-projective. For comparison, the clinical Finnish treebank [4] is reported to have a non-projectivity rate of 2.9% of sentences.

In all sections of the treebank, we have annotated each selected text that is shorter than 75 sentences in its entirety. As for instance some Wikipedia articles may be as long as 300 sentences, longer texts have been truncated after the first 75 sentences to avoid biasing the treebank towards the topics of long texts. This strategy was also used in the construction of the first treebank release.

4 Syntax and Morphology Annotation in the Treebank

4.1 Syntax in the SD Scheme

Our choice for the syntactic representation scheme, the established Stanford Dependency (SD) scheme of de Marneffe and Manning [1, 9], is naturally the same as that used in our previous work. It is a dependency scheme, where syntax is represented as a graph of directed, labeled dependencies between words. The scheme has four different representation variants, which include a different subset of dependency types each. In effect, these variants are layers of dependencies that can be added on top of the basic dependency tree, to offer deeper information on the structure of the sentence. Therefore, the structures in all variants are not necessarily trees. The reader is referred to the original work of de Marneffe and Manning for further details on the SD scheme.

The current annotation of the treebank is based on the so called *basic* variant, where the analyses are trees and the dependencies are for the most part syntactic. The original *basic* variant of the SD scheme includes 55 dependency types, and our modified version 44 types. An example of a syntactic analysis of a Finnish sentence in the SD scheme is given in Figure 1.

In our previous work [3], we have shown that although originally designed for English, the SD scheme is well-suited for Finnish as well, with some minor changes. The reader is referred to this work for details of the modifications made to the original SD scheme, as the version of the scheme used in the current work is identical.

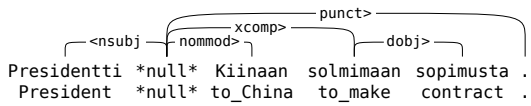

Figure 1: An example of a Finnish sentence annotated in the SD scheme. The sentence can be translated as *The president to China to make a contract*. The *null token* present in the analysis stands for the main verb which this fragmentary sentence lacks but which is necessary for the construction of a dependency analysis in the SD scheme.

word	transl.	lemma	POS	comp.	case	tense	voice	num.	person
Ryöstäjä	Burglar	**ryöstäjä**	**N**		**NOM**			**SG**	
poistui	leave	**poistua**	**V**			**PAST**	**ACT**		**SG3**
pimeän	darkness	**pimeä**	**N**		**GEN**			**SG**	
	dark	pimeä	A	POS	GEN			SG	
turvin	chub	turpa	N		INS			PL	
	safety	**turva**	**N**		**INS**			**PL**	

Figure 2: FinTWOL and FinCG analyses. The words of the sentence and their translations are given in the two leftmost columns and the lemma in the third column, followed by all tags given to the word by FinTWOL. The readings selected by FinCG are shown in bold. The example sentence as read from the leftmost column can be translated as *The burglar left in the safety of the darkness*.

4.2 Morphology with FinTWOL and FinCG

We also add to the whole treebank, including our previously released subcorpus, morphological analyses created using two Finnish morphology tools: FinTWOL and FinCG[3]. For each word, FinTWOL gives all possible readings, each of which includes a detailed morphological analysis. Given the analysis by FinTWOL, FinCG aims to disambiguate which of the readings is correct in the current context. When unable to fully disambiguate a word, FinCG may select multiple readings. In the treebank, each token is given all of its FinTWOL readings, and those selected as correct by FinCG are marked. An illustration of the morphological information present in the treebank is given in Figure 2.

Manually annotated morphological analyses are currently left as future work, pending further investigation of the various issues involved, such as morphological analyzer licensing, defining a suitable annotation scheme and, naturally, funding.

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

Section	Annotator 1	Annotator 2	Annotator 3	Annotator 4	Overall
Wikipedia	95.1	84.0	90.4	-	89.5
Wikinews	96.3	87.7	-	-	92.0
Blogs	94.6	86.4	-	-	90.5
Web-magazine	96.6	89.5	92.0	70.6	88.6
Overall	95.5	86.2	90.8	70.6	**89.9**

Table 2: The inter-annotator agreement of different annotators across the sections of the treebank. All agreement figures are given in labeled attachment scores (%). Note that the LAS is calculated across all tokens and the averages in the table are thus implicitly weighted by the size of the various sections and annotator contributions. Therefore the overall figures are not the same as the averages of the individual annotator or section figures.

5 Annotating the Treebank

5.1 Annotation Process and Quality

Our annotation method for all sections of the treebank is the so called *full double annotation*. Each sentence is first independently annotated by two different annotators, and the resulting annotations are automatically merged into a single analysis, where all disagreements are marked. Disagreements are then jointly resolved, typically by all annotators in the group, and this results in the *merged annotation*. These annotations are further subjected to consistency checks, the purpose of which is to ensure that even old annotations conform to the newest annotation decisions. The result of these consistency checks is called the *final annotation*.

This annotation procedure allows us to measure the quality of the annotation and the suitability of the SD scheme for its purpose, using *inter-annotator agreement*. Rather than the *final annotation*, the agreement is measured for each annotator against the *merged annotation*, so as to avoid unfairly penalizing an annotator on decisions that were correct at annotation time but have later become outdated due to changes in the annotation scheme. Additionally, the *final annotation* may differ from the individual annotations in terms of numbers of tokens and sentences, as sentence boundaries and tokenization are corrected at this level where necessary. We use as the measure of inter-annotator agreement *labeled attachment score (LAS)*, which is the percentage of tokens that receive the correct head and dependency label. On average, our annotators achieved an inter-annotator agreement of 89.9% over the entire treebank. Figure 3 illustrates the development of inter-annotator agreement over time and Table 2 lists the agreements of individual annotators across the different sections of the treebank.

5.2 The Effect of Pre-annotation

On a 45 article subset of the web-magazine section, we have performed an experiment regarding annotation speed and quality, in order to find whether automatic

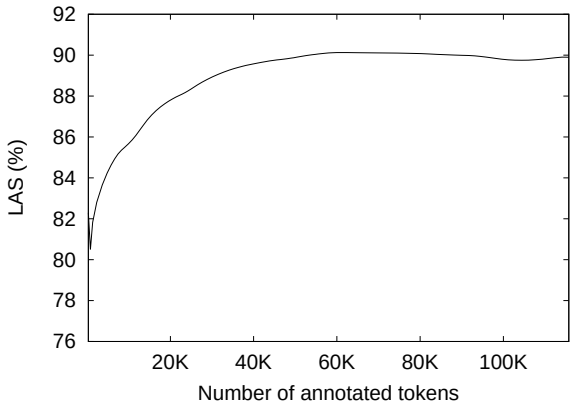

Figure 3: Development of inter-annotator agreement as the amount of annotated tokens grows. Note that the total number of tokens is twice the size of the treebank due to the double-annotation protocol.

Speed [sec/token]				**LAS [%]**			
Annotator	**plain**	**pre.**	***p***	**Annotator**	**plain**	**pre.**	***p***
Annotator 1	3.63	3.01	0.01	Annotator 1	97.4	95.3	<0.001
Annotator 2	4.61	4.45	0.60	Annotator 2	89.0	89.6	0.64
Annotator 3	5.60	5.39	0.65	Annotator 3	92.3	91.7	0.63
Annotator 4	6.92	4.59	0.001	Annotator 4	64.0	78.7	<0.001
Δ		-0.76	<0.001	Δ		2.29	0.034

Table 3: Results of the pre-annotation experiment. The left-hand side shows annotation speed and the right-hand side the LAS. For each annotator are given their base speed, averaged across all plain documents, their pre-annotated speed, and the p-value for the difference. Similarly for LAS. The Δ values are the change of speed or LAS across annotators, corrected for each annotator's base speed or labeled attachment score.

pre-annotation would be helpful (or possibly harmful) for our annotation process. Previously, beneficial effects have been reported by for instance Rehbein et al. [14] and Fort and Sagot [2], on different linguistic annotation tasks.

For the purposes of this experiment, we have produced the first baseline statistical parser of Finnish. Our parser was built using the MaltParser system of Nivre et al. [11], which can be used to automatically induce a parser for a new language, given a treebank. The parser was developed using the body of annotated data available before commencing the experiment discussed in this section, in total 3,648 sentences (48,950 tokens), gathered from all sections of the treebank, including

also a small portion of the web-magazine section. From this data, 80% was used for parser training, 10% for parameter estimation, and 10% for testing. Substantial effort was invested into parameter and feature selection in inducing the parser; the LAS of 70% achieved by the parser thus forms a non-trivial parsing baseline for Finnish. The parser was used to provide automatically pre-annotated versions of each of the documents in our experiment set. The division of documents among annotators was then performed so that for each document, one annotator was assigned the pre-annotated version and the other annotator was to start from an unannotated one (referred to as *plain* hereafter) exactly as in our regular annotation setting. The automatically produced dependencies were visually marked so that the annotator could easily distinguish between dependencies already considered and those still awaiting confirmation or correction.

We calculate the annotation speed in seconds per token and annotation accuracy in terms of LAS. To evaluate the effect of pre-annotation for individual annotators, we compare their speed and LAS on *pre-annotated* vs. *plain* documents and establish statistical significance using the unpaired, two-tailed t-test. The results are shown in Table 3. Our Annotator 4, who has only recently started annotation training and consequently receives the lowest base speed and LAS by far, benefited from the pre-annotation by a tremendous amount, with regard to both speed and LAS. In fact, this annotator's LAS on the plain documents is worse than that of the baseline parser, but given a pre-annotated text, the annotator's LAS clearly exceeds the parser performance. Our most experienced annotator, Annotator 1, gained a small benefit in speed, but suffered a small but statistically significant decrease in LAS. Annotators 2 and 3 did not have a statistically significant difference in speed or LAS.

Since each document was annotated by two different annotators, once as plain and once as pre-annotated, we can further establish the overall effect of pre-annotation across all documents regardless the annotator, using the paired, two-tailed t-test to test for statistical significance. This, however, involves comparing speeds and accuracies between different annotators, which are not directly comparable since individual annotators differ notably in their typical annotation speed and LAS. To take this into account, we establish the base speed and LAS of each annotator across all plain documents (columns *plain* in Table 3) and subtract these from the per-document values before performing the comparison. We are thus comparing changes in speed and LAS, rather than directly their values. We find that pre-annotated documents were on average annotated 0.76 seconds/token faster than plain documents (significant with $p<0.001$) and their LAS was on average 2.29 percentage points higher (significant with $p=0.034$).

Therefore, we conclude that whether pre-annotation is beneficial or harmful depends strongly on the annotator. It would seem that an inexperienced annotator can greatly benefit from a starting point for their work, but for more experienced annotators there was no similar benefit. The risk of overlooking mistakes in a pre-annotated text may contribute to this, and additionally at least Annotator 2 reported difficulties in adapting to the new style and technique of annotation. Naturally, it

could also be suggested that a parser with a better performance could potentially be more helpful for even more experienced annotators. This matter is worth investigating further.

6 Released Data and Software

When releasing the treebank, we do not merely release the text together with its final annotation, but rather the full history leading to the final data. That is, in addition to the final data, we release the independent annotations of both annotators on each document, as well as the *merged* annotation, which is the result of discussing the disagreements between annotators.

Our most important reason for releasing this intermediate data is that each annotated document contains the full edit history of that document, including the exact times (at the resolution of a millisecond) of each edit action performed by an annotator. We believe that this kind of data could potentially be very useful for research, especially for studies on the difficulty of different phenomena encountered in an annotation task, such as the recent work by Tomanek et al. [15]. To our knowledge such detailed data included in a treebank is unique, and it may be a useful resource for future research. In addition, the data makes our own work more transparent. For instance, it allows the replication of the results presented in this paper.

We note that a fraction of approximately 10% of the treebank data in this as well as future releases will be held private, for the purposes of possible future shared tasks on Finnish parsing and parser comparison in general.

Finally, we release a web-based interface for the treebank. This interface allows the user to browse the treebank, as well as make advanced searches. It is possible to search in the text of the treebank, in the morphological analyses, and in the syntactic trees. Morphological and syntactic searches can also be combined, by for instance searching for present tense third singular form verbs that have as their subject a noun that is in partitive. Also searches with a more complex dependency structure are possible, using a syntax akin to TRegex [7] and Tgrep[4]. Detailed documentation of the search features is beyond the scope of this paper and can be found on the project web-page.

7 Conclusion and Future Work

In this work, we have presented an extended version of a freely available treebank for Finnish, the Turku Dependency Treebank (TDT). The size of the current treebank is 4,838 sentences (66,042 tokens), and it consists of four sections, with text from different sources: the Finnish Wikipedia and Wikinews, assorted blogs and a university web-magazine. These sources were selected with the aim to keep the

[4] http://crl.ucsd.edu/software/

treebank freely available under an open license, as well as to ensure a sufficient variation of topics and authors.

The treebank has two levels of analysis: morphological and syntactic. The morphological analyses are created automatically, using existing tools for Finnish. In the manually created syntax annotation, we have used the well established Stanford Dependency scheme, and in order to ensure high quality of the annotation, we have used the *full double annotation* protocol. The average inter-annotator agreement across the treebank was 89.9%. Such a high agreement suggests that annotator training is sufficient and that the annotation scheme is well-defined.

We have also performed an experiment on the effect of pre-annotation on annotation speed and quality and observed greatly improved performance, in terms of both speed and accuracy, for an annotator still in training. An expert annotator achieved a statistically significant gain in speed, although at the cost of a decrease in accuracy.

The treebank, our custom annotation software, detailed data on the annotation process, and a web-based interface of the treebank are available at the address `http://bionlp.utu.fi`.

This work has several important future work directions. The first and most obvious one is to further increase the size of the treebank, adding also new text sources. For instance, fiction text would be a valuable addition, and we are searching for fiction published under an open license. Our current goal is to annotate approximately 10,000 sentences, which appears to generally be enough to produce a robust statistical parser, as for example the results of the multiple language parsing study by Nivre [10] indicate. The second future work direction is to investigate the possibilities to improve the performance of the current parser and to release a fast and robust statistical parser for Finnish.

Thirdly, our goal is to enhance the treebank with additional annotation. Such annotation could, for instance, include human-validated morphological analyses. Also additional dependencies on top of the *basic* SD variant, following one of the extended variants of SD, could be a useful extension of the treebank. With such further annotation in place, it would be possible to add semantic information, for example more detailed analysis of the highly common *nominal modifiers*, with labels such as *temporal* and *cause*. Ultimately, these annotations would enable the development of the treebank into a fully fledged PropBank according to the model set by Palmer et al. [12]. The interaction between the SD and PropBank schemes has already been investigated in connection with the clinical Finnish PropBank [4], and the schemes were found compatible.

Acknowledgements

We are grateful to Lingsoft Ltd. for making FinTWOL and FinCG available to us, as well as the permission to publish their analyses together with the treebank. We would also like to thank all the blog authors who kindly gave us the permission to

include their work in the treebank. This work has been supported by the Academy of Finland.

References

[1] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies manual. Technical report, Stanford University, September 2008.

[2] Karën Fort and Benoît Sagot. Influence of pre-annotation on POS-tagged corpus development. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 56–63, 2010.

[3] Katri Haverinen, Filip Ginter, Veronika Laippala, Timo Viljanen, and Tapio Salakoski. Dependency annotation of Wikipedia: First steps towards a Finnish treebank. In *Proceedings of The Eighth International Workshop on Treebanks and Linguistic Theories (TLT8)*, pages 95–105, 2009.

[4] Katri Haverinen, Filip Ginter, Veronika Laippala, Timo Viljanen, and Tapio Salakoski. Dependency-based propbanking of clinical Finnish. In *Proceedings of The Fourth Linguistic Annotation Workshop (LAW IV)*, pages 137–141, 2010.

[5] Fred Karlsson. Constraint Grammar as a framework for parsing unrestricted text. In *Proceedings of COLING'90*, pages 168–173, 1990.

[6] Kimmo Koskenniemi. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[7] Roger Levy and Galen Andrew. Tregex and Tsurgeon: tools for querying and manipulating tree data structures. In *Proceedings of LREC'06*, pages 2231–2234, 2006.

[8] Mitchell Marcus, Mary Ann Marcinkiwicz, and Beatrice Santorini. Building a large annotated corpus of English: The Penn treebank. *Computational Linguistics*, 19(2):313–330, 1993.

[9] Marie-Catherine de Marneffe and Christopher Manning. Stanford typed dependencies representation. In *Proceedings of COLING'08, Workshop on Cross-Framework and Cross-Domain Parser Evaluation*, pages 1–8, 2008.

[10] Joakim Nivre. Deterministic incremental dependency parsing. *Computational Linguistics*, 34(4):513–553, 2008.

[11] Joakim Nivre, Johan Hall, Jens Nilsson, Atanas Chanev, Gülşen Eryiğit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(2):95–135, 2007.

[12] Martha Palmer, Dan Gildea, and Paul Kingsbury. The Proposition Bank: an annotated corpus of semantic roles. *Computational Linguistics*, 31(1):71–105, 2005.

[13] Sampo Pyysalo, Filip Ginter, Veronika Laippala, Katri Haverinen, Juho Heimonen, and Tapio Salakoski. On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP'07*, pages 25–32, 2007.

[14] Ines Rehbein, Josef Ruppenhofer, and Caroline Sporleder. Assessing the benefits of partial automatic pre-labeling for frame-semantic annotation. In *Proceedings of the Third Linguistic Annotation Workshop*, pages 19–26, 2009.

[15] Katrin Tomanek, Udo Hahn, Steffen Lohmann, and Jürgen Ziegler. A cognitive cost model of annotations based on eye-tracking data. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1158–1167, 2010.

Paper V

A dependency-based analysis of treebank annotation errors.

Haverinen, K., Ginter, F., Laippala, V., Kohonen, S., Viljanen, T., Nyblom, J., and Salakoski, T. (2013). In *Dependency Theory*, Frontiers in Artificial Intelligence and Applications, pages 47–61. IOS Press.

A Dependency-based Analysis of Treebank Annotation Errors

Katri HAVERINEN [a,c,1] Filip GINTER [c] Veronika LAIPPALA [b] Samuel KOHONEN [c]
Timo VILJANEN [c] Jenna NYBLOM [c] and Tapio SALAKOSKI [a,c]
[a] *Turku Centre for Computer Science (TUCS)*
[b] *Department of Language studies, University of Turku*
[c] *Department of Information Technology, University of Turku*

Abstract. In this paper, we investigate errors in syntax annotation with the Turku Dependency Treebank, a recently published treebank of Finnish, as study material. This treebank uses the Stanford Dependency scheme as its syntax representation, and its published data contains all data created in the full double annotation as well as timing information, both of which are necessary for this study.

First, we examine which syntactic structures are the most error-prone for human annotators, and compare these results to those of two baseline parsers. We find that annotation decisions involving highly semantic distinctions, as well as certain morphological ambiguities, are especially difficult for both human annotators and the parsers. Second, we train an automatic system that offers for inspection sentences ordered by their likelihood of containing errors. We find that the system achieves a performance that is clearly superior to the random baseline: for instance, by inspecting 10% of all sentences ordered by our system, it is possible to weed out 25% of errors.

Keywords. Finnish, treebank, annotation, parsing

Introduction

In the field of natural language processing (NLP), human-annotated training data is of crucial importance, regardless of the specific task. The creation of this data requires a large amount of resources, and the data quality affects applications. Thus it is important to ensure that first, the quality of the data is as sufficiently high for the desired purpose, and second, that the amount of expensive manual work is kept to a reasonable amount. Considering the importance of manual annotation for NLP, studies on different aspects of the annotation process are of great interest.

This work strives to examine the difficulty of syntax annotation in the context of Finnish. Our primary objective is to study human annotation and the errors in it, so as to make observations beneficial for future treebanking efforts. As dependency representations have been argued to be a good choice for the purposes of evaluating the correctness of an analysis as well as the general intuitiveness of evaluation measures (see, for instance, the work of Lin [15] and Clegg and Shepherd [2]), and as there ex-

[1] Corresponding Author

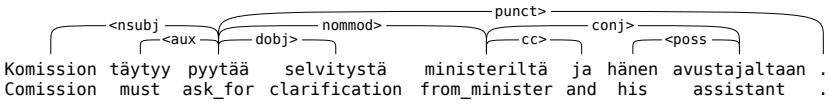

Figure 1. The Stanford Dependency scheme. The sentence can be translated as *The commission must ask for clarification from the minister and his assistant*.

ists a recently published, dependency-based treebank for Finnish, also this study uses dependency-based evaluation.

Our experiments are twofold. First, we conduct an experiment to find which phenomena and constructions are especially error-prone for human annotators. We also compare human errors to those of an automatic baseline parser. Second, as a practical contribution, we build an automatic system that orders annotated sentences in such a way that those sentences most likely to contain errors are presented for inspection first.

The difficulty of annotation is not a heavily studied subject, but there has been some previous work. For instance, Tomanek and Hahn [22] have studied the difficulty of annotating named entities by measuring annotation time. They found that cost per annotated unit is not uniform, and thus suggested that this finding could be used to improve models for active learning [3], the goal of which is to select for annotation those examples that are expected to most benefit an existing machine learning system. Tomanek et al. [23] have conducted a follow-up study using eye-tracking data, and found that annotation time and accuracy depend on both the syntactic and semantic complexity of the annotation unit.

Dligach et al. [6] have studied annotation costs in the context of word sense disambiguation and concluded that for data annotated solely for machine learning purposes, single-annotating a large amount of data appears to be preferable over double-annotating a smaller amount of data. On the level of discourse annotation, Zikánová et al. [24] have examined typical disagreements between annotators in the context of discourse connectives and their scopes, and on the level of syntax, Dickinson [5] has studied the possibilities of finding errors in automatic parses in the context of producing *parsebanks*. More recently, Schwartz et al. [21] have studied the learnability of differently designed syntactic annotation schemes from the point of statistical parsers, and in the context of treebanks, Muhonen and Purtonen [19] have used user intuitions to guide the design of a treebank annotation scheme.

However, studies in the context of manual syntax annotation in particular have been rare. One reason for this may be that data which would enable such studies is not generally available. Many treebanks, such as the well-known Penn Treebank [16], are single-annotated, after an initial annotator training period, and thus agreement of the annotators cannot be measured across the whole treebank. Also, timing data for the annotation process is usually not recorded and made available.

1. Data: The Turku Dependency Treebank

In our experiments, we use the first Finnish treebank, the Turku Dependency Treebank (TDT) by Haverinen et al. [11]. TDT is a treebanking effort still in progress, and the new version used in this work is a superset of the recent second release of the treebank and

consists of 7,076 sentences (100,073 tokens). Approximately 10%[2] of this data is not used in our experiments, except for parser parameter optimization as described below, and this portion of the data will be held secret for the purpose of possible future parser comparisons and scientific challenges. The remaining 90% of TDT, the portion that was used in this work, consists of 6,375 sentences (89,766 tokens). This data is available at the address `http://bionlp.utu.fi/`.

The annotation scheme of the treebank is a slightly modified version of the well-known Stanford Dependency (SD) scheme [17,18]. The original SD scheme is frequently used in applications and due to its popularity, it is to receive updates shortly [4].

The annotation in TDT is based on the *basic* variant of the scheme, in which the analyses are trees of dependencies. In total, the scheme version of Haverinen et al. contains 45 different dependency types, whereas the original scheme version contains 54 types. The scheme modifications include both omissions of types where the corresponding phenomenon does not occur in Finnish, and additions where a phenomenon has not been accounted for in the original SD scheme. Figure 1 illustrates the usage of the SD scheme on a Finnish sentence. In this paper, we only discuss those aspects of the SD scheme that are relevant for the current study. For further details of the scheme, we refer the reader to the annotation manual by de Marneffe and Manning [17], and for changes made during the annotation process of TDT, the manual by Haverinen et al. [9].

The Turku Dependency Treebank is exceptional in the sense that the whole treebank has been created using *full double annotation*, where each sentence is first independently annotated by two different annotators, and all differences are then jointly resolved. This results in a single analysis that is called the *merged* annotation. Afterward, the treebank data is subjected to consistency checks, the purpose of which is to ensure that the final release of the treebank, called the *final* annotation, consists of analyses that are updated to conform to the newest annotation decisions. Consistency checks are needed, as some decisions may need revision when the annotation team comes across new examples, and thus the annotation scheme undergoes slight changes.

The treebank also contains the morphological analyses of two Finnish morphology tools by Lingsoft Ltd., FinTWOL and FinCG [14,13].[3] Out of these, FinTWOL gives each token all of its possible morphological readings, and FinCG disambiguates between these readings. When unable to fully disambiguate, FinCG can select multiple readings.

In addition to the actual treebank — the *final* annotations — TDT releases contain the *individual* annotations of each annotator, two per sentence, and the *merged* annotations. In addition, the documents include a full edit history with millisecond-resolution timestamps.

In total five different annotators have taken part in the annotation of TDT. The annotators have backgrounds including PhD and Master's students in computer science and linguistics, and also their prior knowledge of linguistics varies substantially.

Our experiments have been conducted against the *merged* annotations, not the *final* annotations of the treebank. This is because we want to avoid penalizing an annotator for a decision that was correct at annotation time but has later become outdated. In addition, the numbers of tokens and sentences in the individually annotated documents and in the final treebank documents do not necessarily match, as possible sentence splitting and tokenization issues are corrected at the consistency fix stage of the annotation process. The

[2] 10% on the level of full text documents

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

only exception to this strategy of comparing *individual* annotations against the *merged* annotation is the experiment detailed in Section 3, where an annotator re-annotated some of the treebank sentences, to estimate the quality of the *final* annotation.

For experiments where a baseline parser was needed, we used the MaltParser[4] [20] and the MateTools parser[5] [1]. The former is a transition-based parser, while the latter is a graph-based parser, together representing the two main-stream approaches in statistical dependency parsing. Of the treebank documents, 10% were used for parameter optimization and excluded from the experiments. The remaining portion of the treebank was parsed using *ten-fold crossvalidation*, meaning that 90% of the data was used to train a parser and the remaining 10% was then parsed with it, and this process was repeated ten times in order to parse the whole data (disregarding the parameter optimization set) while ensuring that the training and testing data do not overlap.

2. Error-prone constructions

As the first part of our study, we have examined the numbers of different errors by the human annotators as well as the baseline parser. In these experiments, all dependencies that remain unmatched between the *merged* annotation (henceforth discussed as *gold standard, GS*) and the *individual* annotation (human or automatic), are considered errors. In our measurements, we have used the standard F_1-score, defined as $F_1 = \frac{2PR}{P+R}$, where P stands for precision and R stands for recall. Precision, in turn, is the proportion of correctly annotated dependencies out of all dependencies present in the *individual* annotation, and recall is the proportion of correctly annotated dependencies out of all dependencies present in the gold standard. In some experiments, we also use the *labeled attachment score (LAS)*, the proportion of tokens with the correct governor and dependency type.

In addition to the measurements described below, we have also studied the overall annotator performance on the different *sections*[6] of TDT, in order to find how genre affects the agreement. However, the differences found were small, and it appears that the annotator performance on different genres is similar to the overall performance.

2.1. Most difficult dependency types

In our first set of measurements, we examined which dependency types were the most difficult for the human annotators and the baseline parser. This was done by calculating an F_1-score for each of the dependency types, and the types with the lowest F_1-scores were considered the most difficult ones. Only those dependency types that occur in the gold standard at least 150 times were considered, in order to avoid taking into account types that may have extremely low F_1-scores, but which are also very rare, meaning that their being incorrect hardly affects the overall treebank at all. Table 1 shows the ten most difficult types for the annotators, as well as for the baseline parser.[7]

[4] `http://www.maltparser.org/`

[5] `http://code.google.com/p/mate-tools/`, version 3.3 of November 2012

[6] Current sections include Wikipedia and Wikinews texts, articles from a university online magazine and from student-magazines, blog entries, EU text and grammar examples.

[7] In this experiment, we have disregarded the small single-annotated proportion of TDT constructed in the very beginning of the annotation process in so called *trial annotations*.

Human				
type	**P**	**R**	**F**	**freq.**
iccomp	68.8	70.9	69.8	261 (0.3%)
parataxis	69.9	71.6	70.7	280 (0.4%)
acomp	74.1	70.5	72.2	154 (0.2%)
compar	77.0	71.6	74.2	178 (0.2%)
dep	85.8	69.4	76.7	291 (0.4%)
advcl	79.2	79.1	79.2	982 (1.3%)
auxpass	84.9	75.7	80.0	282 (0.4%)
ccomp	82.2	79.4	80.8	835 (1.1%)
appos	81.7	80.2	81.0	658 (0.9%)
gobj	88.6	77.4	82.6	579 (0.8%)
overall	89.9	89.1	89.5	76,693 (100%)

MaltParser					MateTools				
type	**P**	**R**	**F**	**freq.**	**type**	**P**	**R**	**F**	**freq.**
parataxis	24.2	8.3	12.4	280 (0.4%)	gobj	65.9	26.5	37.8	579 (0.8%)
dep	13.3	34.0	19.1	291 (0.4%)	parataxis	43.0	38.0	40.4	280 (0.4%)
advcl	34.7	39.4	36.9	982 (1.3%)	appos	41.4	40.7	41.0	658 (0.9%)
appos	40.1	38.2	39.1	658 (0.9%)	compar	49.7	44.1	46.7	178 (0.2%)
compar	61.4	35.0	44.6	178 (0.2%)	acomp	52.2	45.5	48.6	154 (0.2%)
acomp	53.2	43.5	47.9	154 (0.2%)	iccomp	51.4	48.8	50.1	261 (0.3%)
rcmod	49.9	48.5	49.2	897 (1.2%)	dep	60.8	43.3	50.6	291 (0.4%)
ccomp	57.0	49.4	52.9	835 (1.1%)	advcl	54.4	49.6	51.9	982 (1.3%)
iccomp	63.7	48.0	54.8	261 (0.3%)	nn	62.1	55.8	58.8	695 (0.9%)
conj	61.0	63.0	62.0	4,041 (5.3%)	ccomp	63.9	62.9	63.4	835 (1.1%)
overall	71.5	71.2	71.3	76,693 (100%)	**overall**	74.4	74.7	74.6	76,693 (100%)

Table 1. The ten hardest dependency types for human annotators and the two parsers. The standard F_1-score was calculated for each dependency type separately, considering only those types that occur in the gold standard at least 150 times. This table presents the ten dependency types with the lowest F_1-scores. For each type is given its precision, recall and F_1-score, and its frequency in the gold standard.

From this table it can be seen that several of the most difficult dependency types for human annotators represent a complement of the verb. The annotation scheme of the treebank contains several different types for these complements, such as clausal complement (*ccomp*) and infinite clausal complement (*iccomp*), as well as a clausal complement with external subject (*xcomp*). Distinguishing these types, especially *ccomp* and *iccomp*, is often challenging, as the distinction depends on only the form of the complement verb. Adjectival complements (*acomp*) likely fall victim to the difficulty of assessing whether a sentence element is a complement. The attachment of sentence elements can also be a source of difficulty. For instance, in word order variations of an example like *The man in the brown coat came into the train* it may be difficult to determine whether *in the brown coat* should modify *man* or *came into the train*. In these cases, the analysis in the treebank follows rules similar to those used in the Prague Dependency Treebank [7], where in *The man in the brown coat came into the train* there is considered to be a *man in the brown coat*, but in *The man came into the train in a brown coat* the coming into the train happened while wearing a brown coat. These rules, however, are easy to overlook especially in fast-paced annotation. Adverbial clause modifiers (*advcl*), non-complement subordinate clauses, are an example of a phenomenon where the difficulty of annota-

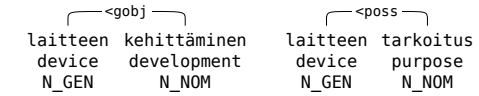

Figure 2. Genitive objects (left) and other genitive modifiers (right). The examples can be translated as *the development of the device* and *the purpose of the device*, respectively. The word *laitteen* (genitive form of *device*) is a genitive attribute of the noun in both examples, but on the left, the noun *kehittäminen* has been derived from the corresponding verb *kehittää* (*to develop*), and the device acts as the object of the developing. Direct derivations of a verb are morphologically marked in the treebank, but other verb-related nouns are not.

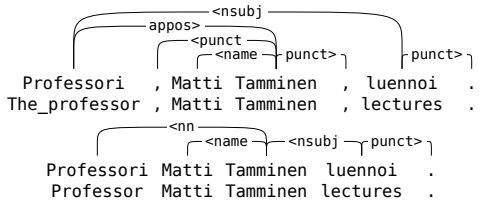

Figure 3. Appositions (top) and appellation modifiers (bottom). The examples can be translated as *The professor, Matti Tamminen, lectures* and *Professor Matti Tamminen lectures*, respectively. The key difference between the examples is that the apposition structure includes commas, while the one with the appellation modifier does not.

tion may be partly due to attachment issues and partly the difficulty of distinguishing complements and modifiers.

The dependency type *parataxis* is used to mark two different phenomena: direct speech and certain types of implicit clausal coordination, for instance, clauses combined using a semicolon. Especially the latter use can be difficult due to the phenomenon being closely related to coordination. Comparative structures (marked with the type *compar*), in turn, are often elliptical, and it may be unclear what is being compared with what.

Passive auxiliaries (*auxpass*) may suffer from the annotator simply forgetting them, as there is also a more general dependency type for auxiliaries (*aux*). In some cases drawing the line between passives and other subjectless expressions[8] may be difficult. In addition, some passive participles can also be interpreted as adjectives, and thus clauses containing these participles can be read as copular. Another mistake that is easily made out of carelessness is that of mistaking genitive objects (*gobj*) for more general genitive modifiers (*poss*). On the other hand, the distinction of genitive objects and general genitive modifiers is also highly semantic in nature. For an illustration of genitive objects in the SD scheme, see Figure 2.

Another difficult phenomenon seen in Table 1 is the apposition (*appos*). Appositions are often hard to distinguish from nominal modifiers (*nommod*) due to the semantic requirement that an apposition should have the same referent as its head word. In addition, the annotation scheme distinguishes between appositions and appellation modifiers (marked with the type *nn* alongside with noun compound modifiers), where the distinction usually depends on small details such as the inflection forms of the words involved or the presence or absence of punctuation. Figure 3 illustrates appositions and appellation modifiers in the Finnish-specific version of the SD scheme. Finally, the most

[8] such as the *zeroth person*, *nollapersoona* [8, §1347]

generic dependency type *dep* (dependent) is also among the most difficult types. This type is meant for cases where no other, more specific type applies, and in the treebank, it is mostly used for idiomatic two-word expressions.

The most difficult dependency types for the automatic parsers are in some respects similar compared to humans, although there are differences as well. Like human annotators, both parsers had difficulties with different clausal complements and modifiers (types *ccomp*, *advcl* and *iccomp*), and unlike humans, MaltParser also scored low on relative clause modifiers (*rcmod*). Appositions were also clearly difficult for both parsers, which is understandable due to the semantic distinctions involved. Another two types that were difficult for the parsers but not particularly for humans, were *conj* (coordinated element, see Figure 1) for MaltParser and *nn* for MateTools. With coordinations, it is difficult for a parser to decide which sentence element is coordinated with which, and additionally, for instance an apposition structure may seem coordination-like without any semantic information. The closely related *parataxis* was also especially difficult for the parsers. The low F_1-score of the type *nn*, which is used to mark noun compounds and appellation modifiers, may have to do with its close relation to the *name* and *appos* types. When comparing the two parsers, eight of the ten most difficult dependency types are shared. Even though their individual ranking and performance differ to some extent, these dependency types seem difficult for parsers regardless of the underlying parsing paradigm. The overall F_1-scores of the parsers are 71.3% and 74.6%, respectively, which is considerably lower than the human performance of 89.5%.

2.2. Dependency type confusions

Seeing that for many of the most difficult dependency types, the potential explanation seemed to include a possible confusion with another type, we have investigated this matter further. We have calculated the numbers of those errors where the governor is correct, but where the dependency type is wrong, that is, where a dependency type has been replaced by another type. Table 2 shows the five most common type confusions for all five annotators as well as the parser. In total, approximately 32.4% of all erroneous dependencies assigned by annotators only had an incorrect dependency type.

The confusion errors can be divided into several different classes. One error type that can be seen from the table are errors arising from both morphological and semantic closeness of two phenomena. For instance, a common type confusion for nearly all annotators was that of confusing the types *nommod (nominal modifier)* and *dobj (direct object)*. The distinction between nominal modifiers and direct objects is based on both structure and morphology; objects are complements of the verb that can only take certain cases of the Finnish case system [8, §925]. It is likely that the semantic closeness of objects and certain nominal modifiers misled annotators. In addition, some measures of amount take the same cases as objects and closely resemble them. A nominal modifier like this is called an *object-cased amount adverbial*[9] [8, §972].

Also a second confusion seemed to be affected by morphological and semantic closeness. This confusion occured particularly for Annotators 2 and 4, who notably confused subjects and objects on occasion. For other annotators this confusion occured as well, but not as frequently. Subjects and objects may at first seem like a surprising confusion pair, but actually, due to several reasons these two can rather easily be confused in

[9] *objektin sijainen määrän adverbiaali (OSMA)*

Annotator 1			Annotator 2			Annotator 3		
GS type	**annot. type**	**fr. (%)**	**GS type**	**annot. type**	**fr. (%)**	**GS type**	**annot. type**	**fr. (%)**
advmod	nommod	5.6	dobj	nommod	6.8	advmod	nommod	6.8
dobj	nommod	3.7	gobj	poss	5.5	dobj	nommod	5.7
auxpass	aux	3.0	nsubj	dobj	4.8	nommod	dobj	4.2
gobj	poss	2.9	advmod	nommod	4.4	advmod	advcl	3.0
nommod	advmod	2.6	nommod	dobj	4.3	nommod	appos	3.0

Annotator 4			Annotator 5		
GS type	**annot. type**	**fr. (%)**	**GS type**	**annot. type**	**fr. (%)**
dobj	nommod	11.5	dobj	nommod	7.1
nommod	dobj	6.0	acomp	nommod	7.1
gobj	poss	5.4	partmod	advcl	5.4
nsubj	nommod	5.1	appos	conj	5.4
nsubj	dobj	4.0	nommod	dobj	5.4

MaltParser			MateTools		
GS type	**annot. type**	**fr. (%)**	**GS type**	**annot. type**	**fr. (%)**
gobj	poss	4.8	gobj	poss	9.3
nommod	dobj	4.7	nommod	dobj	5.1
partmod	amod	4.4	dobj	nsubj	4.7
name	poss	3.9	name	poss	4.5
dobj	nsubj	3.6	dobj	nommod	4.2

Table 2. The five most common dependency type confusions for each annotator and the two parsers. For each confusion is given the gold standard dependency type (GS type) and the type suggested by the annotator (annot. type), as well as the frequency of the confusion, out of all type confusions by the annotator/parser.

Finnish, especially when annotating quickly. First, both subject and object use the same cases of the Finnish case system: the nominative, the partitive, and the genitive. Second, Finnish is a free word-order language, and thus the word-order does not necessarily reveal the role of a word. Also, certain verbs that are passive-like in nature, but in fact take a subject and not an object, so called *derived passives* [10] [8, §1344], further add to the misleading characters of subjects and objects. In the majority of cases, it is not difficult to decide which of the two analyses is correct in the annotation scheme, once the disagreement is brought into attention, but rather it is the case that annotators are easily misled by the similar properties of these two sentence elements.

A second error type seen in the table is a confusion that is based on a difficult morphological distinction. The distinction between nominal (*nommod*) and adverbial modifiers (*advmod*) was, for several annotators, among the most difficult ones. It is not always clear whether a word should be analyzed as an adverb or rather as an inflected noun, as it is possible for many adverbs to inflect in certain cases, similarly to nouns. For instance, the Finnish word *pääasiassa* (*mainly*) could be analyzed as an adverb, or it could be seen as an inflected form of the noun *pääasia* (*main thing*).

One unexpected type of confusion errors was typical for Annotator 3 in particular. These errors are not due to linguistic similarity, but are simply typographical errors. The annotator has confused adverb modifiers (*advmod*) with adverbial clause modifiers (*advcl*), which are linguistically rather easily distinguishable, but in the annotation software user interface, the shortcut key for *advmod* is capital *V*, while the shortcut key for *advcl* is non-capital *v*. Similarly, this annotator has confused also other dependency types where the respective shortcut keys were capital and non-capital versions of the same letter, but these were not as frequent. Annotator 1 also used the shortcut keys of the annotation user interface and made some typographical errors, although not frequently enough to appear among the five most common type confusions. An example of such an error by Annotator 1 is the confusion of subjects (*nsubj*) and adjectival modifiers (*amod*). The

[10] *johdospassiivi*

explanation for this otherwise peculiar error is that the shortcut key for *nsubj* is *s* and the one for *amod* is *a*, which are adjacent on a regular Finnish keyboard.

The automatic parsers also displayed confusion errors in their output (approximately 17.1% and 20.9% of all erroneous dependencies for MaltParser and MateTools respectively), involving many of the same semantic distinctions that were difficult for humans, such as genitive objects versus other genitive modifiers and nominal modifiers versus direct objects. Notably the confusion of subjects and objects was also present. Also one morphological distinction was among the most difficult ones for the parsers: participal versus adjectival modifiers, where the distinction is, in fact, between participles and adjectives. This confusion error is 3rd most common for MaltParser and 7th most common for MateTools. The same confusion was present for human annotators, but not among the five most common ones. As an example, consider the Finnish word *tunnettu* (*well-known*). It could be a form of the verb *tuntea* (*to know*), but on the other hand, it can be given the comparative and superlative forms, which are typical of adjectives. The only type of confusions that did not, naturally, occur for the parser were the typographical errors. Similarly as in Table 1, the two parsers behave alike, with four out of the five most common confusion errors being shared.

2.3. Correlation of human and parser errors

An interesting question to study is whether the annotator and parser errors correlate with respect to their position in the sentence. Such correlation would indicate that certain structures are in some sense "universally difficult", regardless of whether the annotator is human or machine. This correlation is easy to analyze on the level of tokens: a token is deemed correct if its governor and dependency type are correct. Since we have two independent human annotations for each sentence, we take the union of the individual annotators' errors, thus defining a token as correct only if it was correctly analyzed by both of the annotators. In this experiment, we can only take a sentence into account, if it has both human analyses available. This is the case for a total of 82,034 tokens not used for parser optimization, as a small portion of TDT has, in the very beginning of the annotation process, been constructed in so called *trial annotations*, where a single annotator has annotated the sentence and it has then been jointly inspected [10].

The results are shown in Table 3. For MaltParser, we find that 36.3% (8,433/23,202) of parser errors co-occur with human errors, whereas only a 19.0% (15,627/82,034) co-occurrence, corresponding to the human error-rate, would be expected by chance. Similarly, we find that 53.9% (8,433/15,627) of human errors co-occur with parser errors, whereas only a 28.2% (23,202/82,034) co-occurrence, corresponding to the parser error-rate, would be expected by chance. For MateTools, closely matching values are seen and similar observations can be made. Both for MaltParser and MateTools, there thus a notable positive association between human and parser errors, strongly statistically significant with $p \ll 0.001$ (Pearson's chi-square test on Table 3).

Finally, in Table 4, we analyze the correlation of errors between the two parsers. Well in line with the results in the previous sections, we find very high correlation between the errors of the two parsers, with full 60.2% (13,959/23,202) of MaltParser errors and 69.3% (13,959/20,143) of MateTools errors shared, compared to 28.3% (23,202/82,034) and 24.6% (20,143/82,034) expected by chance. This correlation is strongly statistically significant with $p \ll 0.001$ (Pearson's chi-square test on Table 4).

		human	
		error	correct
MaltParser	error	8,433	14,769
	correct	7,194	51,638
MateTools	error	7,554	12,589
	correct	8,073	53,818

Table 3. Token-level correlation between human and parser errors.

		MateTools	
		error	correct
MaltParser	error	13,959	9,243
	correct	6,184	52,648

Table 4. Token-level correlation between MaltParser and MateTools.

3. Correctness of double-annotated data

As part of our investigation on the number of errors by human annotators, we have conducted a small-scale experiment on the correctness of the final treebank annotation. We sampled a random set of 100 sentences from the *final* annotations of the treebank and assigned them to an annotator who had not annotated them previously. This annotator then independently re-annotated these sentences, and the resulting annotation was compared to the previously existing *final* annotation in a regular meeting between all the annotators.

Effectively, we thus gained a set of triple-annotated sentences. The *final* annotation of the corresponding portion of the treebank was compared against these triple-annotated sentences, and thus we gained an estimate of the error-rate of the *final* annotation in the treebank. The LAS for the final treebank annotation against the triple-annotated sample as gold standard was 98.1%, which means that the minimum error-rate of the *final* annotation is 1.9%. This is a lower bound, as it is possible (although unlikely) that further errors go unnoticed because three annotators have given a sentence the same, erroneous analysis.

We thus find that *full double annotation* is an efficient way to produce annotation of high quality. The triple annotation agreement of 98.1% together with the original inter-annotator agreement of 89.6% in LAS (89.5% in $F_1 - score$) implies that approximately 82% ((98.1-89.6)/(100-89.6)) of errors remaining in the single annotated documents can be weeded out using double annotation.

4. Automated recognition of annotation errors

While full double annotation produces high-quality results, as shown in the previous section, it is undoubtedly a resource-intensive approach to annotation. In many cases, particularly when building large treebanks, a compromise between single and double annotation will be necessary. Under such a compromise annotation strategy, only some proportion of sentences would be double annotated or otherwise carefully inspected for errors, while the rest would remain single-annotated. If we were to select these sentences randomly, we would expect to correct the same proportion of annotation errors present in the treebank, assuming that the errors are approximately evenly distributed throughout

the treebank. Thus, for example, by randomly selecting 25% of the sentences for double annotation we would expect to visit 25% of annotation errors present in the treebank. The necessary effort would naturally decrease if we used a strategy that is better than random at selecting sentences which contain annotation errors. In the following, we investigate a machine-learning based method which, given a single-annotated sentence, assigns each token a score that reflects the likelihood of that token being an annotation error, i.e., not having the correct governor and dependency type in the tree.

We approach the problem as a supervised binary classification task where incorrectly annotated tokens are the *positive class* and correctly annotated tokens are the *negative class*. Training data for the classifier can be obtained from the individual annotators' trees, by comparing them against the *merged* trees resulting from the double annotation protocol. If for any token its governor or dependency type do not match those in the merged tree, this token is considered an annotation error (a positive instance), otherwise it is considered correct (a negative instance). Since the average LAS of our annotators is about 90%, the training data contains about 10% positive instances and 90% negative instances, a considerably disbalanced distribution.

The features that represent the tokens in classification are as follows:

Annotator The annotator who produced the tree.

Morphology/POS The lemma, POS, and morphological tags given for all possible morphological readings of the word (prefixed by "cg_" if the reading was selected by the FinCG tagger). The number of possible morphological readings of the word, and the number of readings selected by FinCG.

Dependency Whether the token acts as a dependent, the dependency type, and all morphology/POS features of the governor, given both separately and in combination with the dependency type. The same features are also generated for all dependents of the token under consideration.

We split the available data randomly into a training set (50%), a parameter estimation set (25%), and a test set (25%). The split is performed on the level of documents, so that all instances generated from both annotations of a single document are always placed together in one of the three sets. This prevents any possible leak of information between data used for training and that used for testing. As the classifier, we use the support vector machine (SVM)[11] [12] with the radial basis kernel. We select the C and γ parameters by a wide grid search on the parameter estimation set. To account for the pronounced disbalance in the positive and negative class distribution, we use the standard *area under ROC curve (AUC)* performance measure, which is not sensitive to class distribution, and is thus preferred in this case over the usual F_1 or accuracy. We use AUC as both the SVM loss function and the performance criterion to select the best parameter combination.

To evaluate the accuracy of the classification, and its practical impact on annotation, we first calculate for each sentence the maximum of the classification scores over all of its tokens and then order the sentences in descending order by this value. The results on the test set are shown in Figure 4. The classifier is notably better than the random baseline: the first 10% of the sentences contain 25% of all annotation errors, and the first 25% of the sentences contain 50% of all annotation errors. These differences are large enough to provide a notable decrease in annotation effort. For instance, the effort to correct 50%

[11] Implemented in the *SVM^{perf}* package available at `http://www.joachims.org`

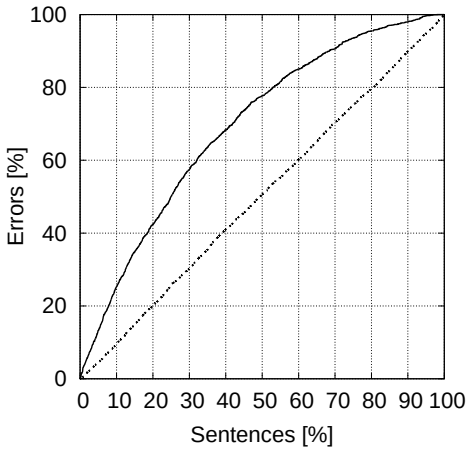

Figure 4. Proportion of annotation errors recovered. The full line represents the machine-learning based ordering of sentences, while the dashed line represents a baseline obtained by ordering the same sentences randomly.

of annotation errors is halved: only 25% of all sentences need to be double-annotated, instead of the 50% random baseline. For a treebank of 10,000 sentences, this would mean that 2,500 sentences less would need to be double annotated, a notable reduction of effort. Here it should be noted that the classification problem is relatively difficult: we are asking a classifier to recognize human mistakes, at a task at which the humans are 90% correct to start with.

We have also investigated, as an additional feature for the classification, the time spent by the annotator to insert all dependencies for the given token (governor and all dependents), including dependencies that are removed or relabeled in the course of the annotation. Our hypothesis was that those parts of the sentence on which the annotator spent an unusually long time are more difficult to analyze, and thus prone to error as well. This experiment is possible since the treebank contains annotation history data with millisecond-resolution timestamps. However, a substantial part of the treebank is annotated so that of the two individual annotations for each sentence, one is constructed from scratch with all dependencies inserted manually, while the other is constructed on top of the output of a parser, with the annotator correcting the parser output [11]. Complete timing data can naturally be extracted only in the former case, amounting to 119,117 tokens.

We further normalize the annotation timing data to account for the different baseline annotation speeds of the annotators, as well as for the simple fact that the annotation of a token with more dependencies takes longer to complete. We first divide the annotation time of each token by the number of its dependencies in the completed tree and then, for each sentence separately, subtract from each time the mean and divide by standard deviation of the times in that particular sentence. Thus normalized annotation times were then included as a feature in the classification. However, no measurable gain in the performance of the classifier could be observed.

	correct	incorrect
slow	14,752	2,288
normal	92,290	9,787

Table 5. Correlation between annotation speed and correctness of tokens. Tokens are defined as "slow" if their annotation took longer than one standard deviation above the mean time.

To investigate the correlation between annotation speed and annotation accuracy further, we define a token as "slow" if the time it took to complete is more than one standard deviation above the mean[12] time in the given sentence (we first divide by the number of the token's dependencies, as previously). We then correlate the correctness and speed of annotation in a contingency table (Table 5). We find that incorrectly annotated tokens are overrepresented among "slow" tokens (13.4%), compared to the rest of the tokens (9.6%), as per our original hypothesis. This positive association is strongly statistically significant ($p \ll 0.001$, Pearson's chi-square test on Table 5). While this observation is of some interest, the magnitude of the difference is likely too small for practical applications and annotation times do not seem to provide new information — on top of the features listed above — to a classifier predicting incorrectly annotated tokens.

5. Conclusions and future work

In this paper, we have studied the difficulty of syntax annotation in a dependency-based framework, in the context of the Finnish language and the Stanford Dependency (SD) scheme. We have studied the different kinds of errors by the annotators and compared these errors with those of two baseline parsers. In addition, we have trained an automatic system that orders single-annotated sentences so that sentences that are most likely to contain errors are offered for inspection first.

We find that there are several different kinds of mistakes that humans make in syntax annotation. In this data, different kinds of clausal complements and modifiers were often erroneously marked, as were comparatives, appositions and structures with parataxis. Nearly one third of the erroneous dependencies marked by annotators were such that only the type of the dependency was wrong. Morphological and semantic closeness of two phenomena seemed to mislead annotators, as for instance adverbial modifiers were often confused with nominal modifiers, and nominal modifiers with direct objects. Annotators also made some mistakes that were not due to any linguistic resemblance, but rather an artifact of annotation user interface shortcut keys that were adjacent or capital and non-capital versions of the same letter. The last type of errors suggests how this particular annotation user interface in question could be improved, or how the usability of possible future software could be increased.

We also find that our automatic sentence ranker notably outperforms a random baseline. This means that using this classifier to order single annotated sentences for inspection, it is possible to significantly reduce the amount of double annotation or other careful inspection needed in a compromise setting where full double annotation is not possible or desired. For instance, if one wanted to correct 50% of errors in a treebank, using the proposed method, they could inspect only 25% of all sentences instead of the 50% expected by random selection — a remarkable decrease in effort.

[12] Variations of this definition were tested and had no effect on the overall conclusion.

In the future, the knowledge gained in this work could be used for developing new methods helpful for inspecting manual annotations, and for the benefit of large annotation efforts in general. Also studies in for instance the field of active learning, where the goal is to keep the amount of data annotated for machine learning purposes to a minimum, could be conducted.

Acknowledgments

We would like to thank Lingsoft Ltd. for their kind permission to use FinTWOL and FinCG analyses in TDT. We are also grateful to the corpus text authors for the use of their text. This work has been supported by the Academy of Finland and the Emil Aaltonen foundation. Computational resources were provided by CSC – IT Center for Science.

References

[1] Bernd Bohnet. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of COLING'10*, pages 89–97, 2010.

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

[3] D. Cohn, Z. Ghahramani, and M. Jordan. Active learning with statistical models. *Journal of Artificial Intelligence Research*, 4:129–145, 1996.

[4] M.-C. de Marneffe, M. Connor, N. Silveira, S. Bowman, T. Dozat, and C. D. Manning. More constructions, more genres: Extending Stanford Dependencies. In *Proceedings of Depling'13*, 2013. To appear.

[5] M. Dickinson. Detecting errors in automatically-parsed dependency relations. In *Proceedings of ACL'10*, pages 729–738, 2010.

[6] D. Dligach, R. Nielsen, and M. Palmer. To annotate more accurately or to annotate more. In *Proceedings of the Fourth Linguistic Annotation Workshop*, pages 64–72, 2010.

[7] J. Hajič. Building a Syntactically Annotated Corpus: The Prague Dependency Treebank. In *Issues of Valency and Meaning. Studies in Honour of Jarmila Panevová*, pages 106–132. Karolinum, Charles University Press, Prague, Czech Republic, 1998.

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

[9] K. Haverinen. Syntax annotation guidelines for the Turku Dependency Treebank. Technical Report 1034, Turku Centre for Computer Science, January 2012.

[10] K. Haverinen, F. Ginter, V. Laippala, T. Viljanen, and T. Salakoski. Dependency annotation of Wikipedia: First steps towards a Finnish treebank. In *Proceedings of TLT8*, pages 95–105, 2009.

[11] K. Haverinen, T. Viljanen, V. Laippala, S. Kohonen, F. Ginter, and T. Salakoski. Treebanking finnish. In *Proceedings of TLT9*, pages 79–90, 2010.

[12] T. Joachims and C-N. John Yu. Sparse kernel SVMs via cutting-plane training. *Machine Learning, Special Issue from ECML PKDD 2009*, 76(2–3):179–193, 2009.

[13] F. Karlsson. Constraint Grammar as a framework for parsing unrestricted text. In *Proceedings of COLING'90*, pages 168–173, 1990.

[14] K. Koskenniemi. Two-level model for morphological analysis. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 683–685, 1983.

[15] D. Lin. A dependency-based method for evaluating broad-coverage parsers. *Natural Language Engineering*, 4(2):97–114, 1998.

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

[17] M-C. de Marneffe and C. Manning. Stanford typed dependencies manual. Technical report, Stanford University, September 2008.

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

[19] K. Muhonen and T. Purtonen. Creating a dependency syntactic treebank: Towards intuitive language modeling. In *Proceedings Depling'11*, pages 155–164, 2011.

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

[21] R. Schwartz, O. Abend, and A. Rappoport. Learnability-based syntactic annotation design. In *Proceedings Coling'12*, pages 2405–2422, 2012.

[22] K. Tomanek and U. Hahn. Annotation time stamps — temporal metadata from the linguistic annotation process. In *Proceedings of LREC'10*, pages 2516–2521, 2010.

[23] K. Tomanek, U. Hahn, S. Lohmann, and J. Ziegler. A cognitive cost model of annotations based on eye-tracking data. In *Proceedings of ACL'10*, pages 1158–1167, 2010.

[24] Š. Zikánová, L. Mladová, J. Mírovský, and P. Jínová. Typical cases of annotators disagreement in discourse annotations in prague dependency treebank. In *Proceedings of LREC'10*, pages 2002–2006, 2010.

Paper VI

Building the essential resources for Finnish: the Turku Dependency Treebank.

Haverinen, K., Nyblom, J., Viljanen, T., Laippala, V., Kohonen, S., Missilä, A., Ojala, S., Salakoski, T., and Ginter, F. (2013). *Language Resources and Evaluation*. In press. Available online. DOI: 10.1007/s10579-013-9244-1.

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.

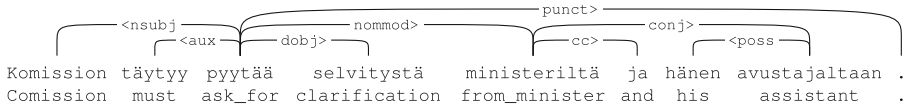

Fig. 1 The stanford dependency scheme. The sentence can be translated as *The commission must ask for clarification from the minister and his assistant*

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

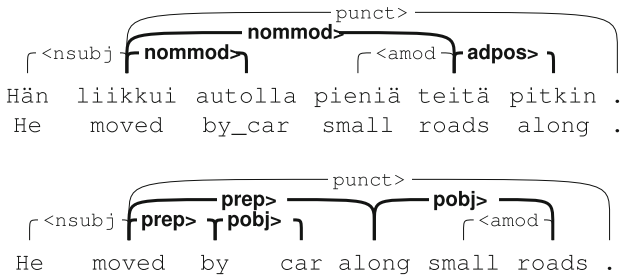

Fig. 2 Nominal modifiers and adpositions in Finnish (*top*). Note how in adpositional phrases, the nominal is made the governor of the phrase. The sentence can be translated as *He moved by car along small roads*. For comparison, also the corresponding English sentence and its analysis in the original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

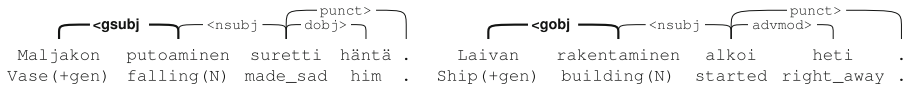

Fig. 3 Genitive subjects (*left*) and objects (*right*) of a noun. The examples can be translated as *The falling of the vase made him sad* and *The building of the ship started right away*

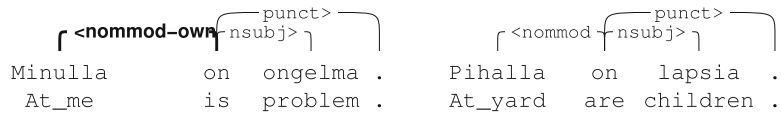

Fig. 4 A possessive clause (omistuslause, *left*) as compared to an existential clause (*right*). Otherwise the analysis is exactly the same, but in the possessive clause the owner is marked using the dedicated dependency type *nommod-own*. The examples can be translated as *I have a problem* for the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

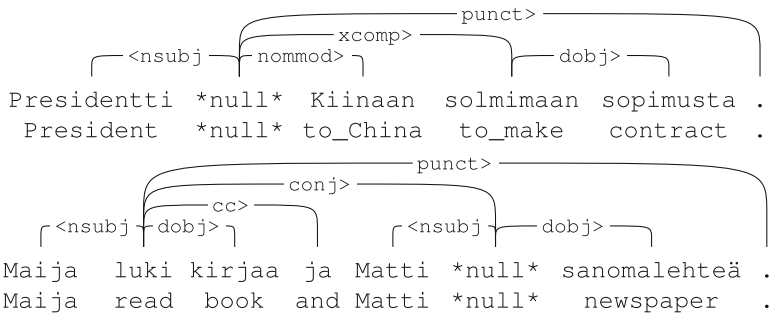

Fig. 5 Fragments (*top*) and gapping (*bottom*). When a word necessary for constructing an analysis is missing, a *null token* is inserted to represent it. The fragment example can be translated as *The president to China to make a contract* and the gapping example as *Maija read a book and Matti a newspaper*

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

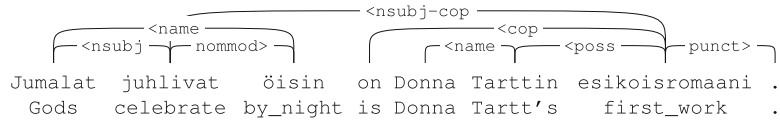

Fig. 6 Multi-word named entities are marked with the dependency type *name*. Note how *Jumalat juhlivat öisin* has an internal structure (*"Gods celebrate by night"*, the Finnish title of *A secret history*), where the main verb, *juhlivat*, acts as the governor, whereas *Donna Tartt* is merely a name.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

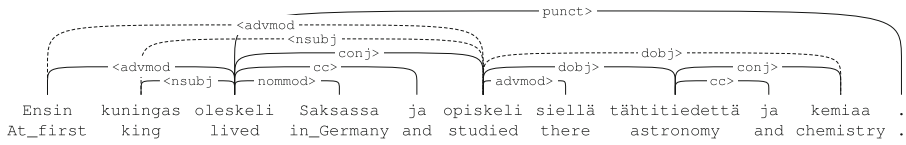

Fig. 7 Propagation of conjunct dependencies. The base layer of annotation is marked with solid dependencies, and the propagated dependencies are dashed. The example sentence can be translated as *First the King lived in Germany and studied there astronomy and chemistry*. Note how the subject (*kuningas*, *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

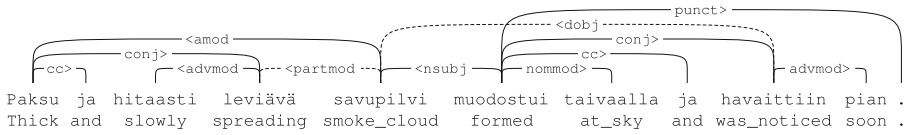

Fig. 8 Propagation of conjunct dependencies with dependency type changes. On the *left*, the adjectival modifier is coordinated with a participal modifier, and thus the type of the *amod* dependency changes into *partmod* while propagating. On the *right*, the word *savupilvi* (*cloud of smoke*) acts 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

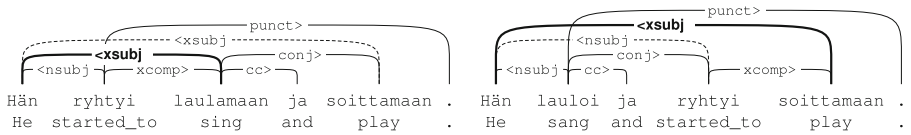

Fig. 9 External subjects. Note how the *xsubj* dependencies can propagate (*left*), and how propagated subjects can produce an external subject (*right*). The examples can be translated as *He started to sing and play* and *He sang and started to play*

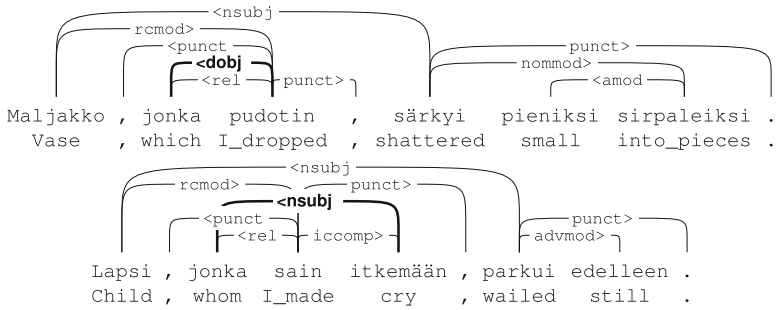

Fig. 10 Syntactic functions of relativizers. A relativizer can act in any syntactic function, such as an object (*top*) or a subject (*bottom*). Note that in the bottom-most example, the *nsubj* dependency does not coincide with the *rel* dependency that it corresponds to on the first layer. The examples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

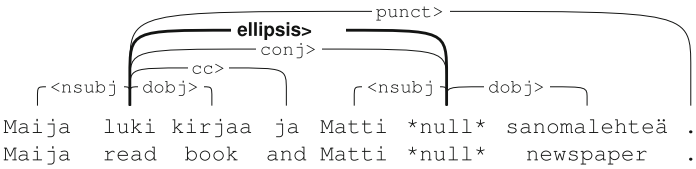

Fig. 11 Gapping is marked by a null token to represent the elided word, as well as a dependency of the type *ellipsis*. The example can be translated as *Maija read a book and Matti a newspaper*

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	isolisä	grandlfather	A + N	Pos Sg Nom Cmpnd + Sg Nom
	isolisä	grandlfather	N + N	Pfx Cmpnd + Sg Nom
	isolisä	grandlfather	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.

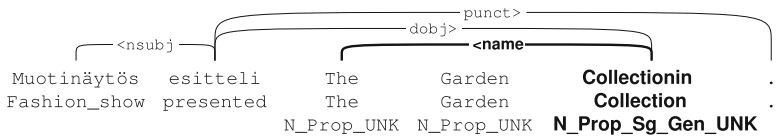

Fig. 12 Unknown words under a *name* dependency can inherit the morphological analysis of the governor, given that there is no internal structure annotated for the named entity. Note how only the main categories, N (noun) and Prop (proper) are inherited, and tags signaling less important features such 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

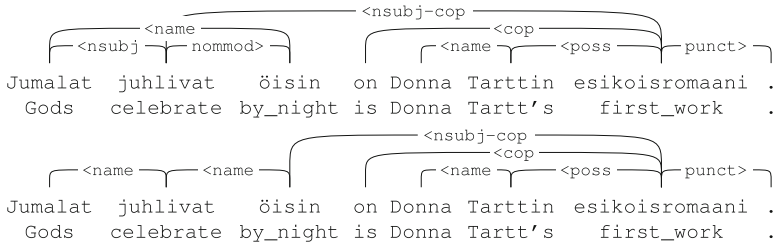

Fig. 13 The original *name* dependencies in TDT (*top*), and the *name* dependencies as processed in order to ensure treeness (*bottom*)

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

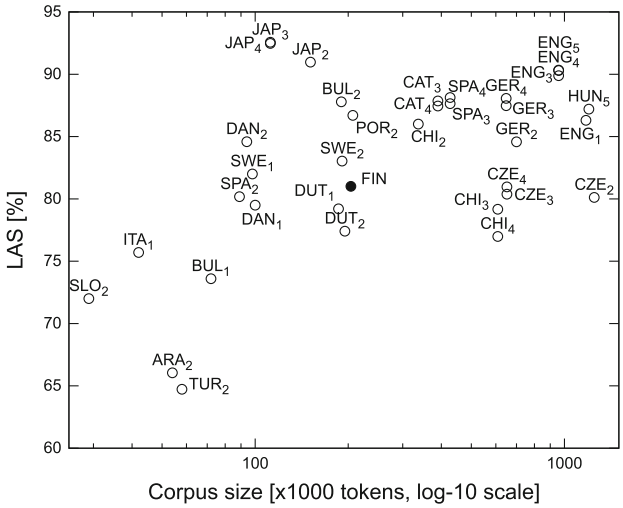

Fig. 14 Parsing results for various languages from a number of studies, as related to corpus size. The languages are given as short labels, where *CAT* Catalan, *CHI* Chinese, *CZE* Czech, *ENG* English, *GER* German, *JAP* Japanese, *SPA* Spanish, *ARA* Arabic, *BUL* Bulgarian, *DAN* Danish, *DUT* Dutch, *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.

Paper VII

Towards a dependency-based PropBank of general Finnish.

Haverinen, K., Laippala, V., Kohonen, S., Missilä, A., Nyblom, J., Ojala, S., Viljanen, T., Salakoski, T., and Ginter, F. (2013) Towards a dependency-based PropBank of general Finnish. In *Proceedings of NODALIDA'13*, pages 41–57.

Towards a Dependency-based PropBank of General Finnish

Katri Haverinen,[1,2] *Veronika Laippala,*[3] *Samuel Kohonen,*[2] *Anna Missilä,*[2]
Jenna Nyblom,[2] *Stina Ojala,*[2] *Timo Viljanen,*[2]
Tapio Salakoski[1,2] *and Filip Ginter*[2]

(1) Turku Centre for Computer Science (TUCS), Turku, Finland
(2) Department of Information Technology, University of Turku, Finland
(3) Department of Languages and Translation Studies, University of Turku, Finland

first.last@utu.fi

ABSTRACT
In this work, we present the first results of a project aiming at a Finnish Proposition Bank, an annotated corpus of semantic roles. The annotation is based on an existing treebank of Finnish, the Turku Dependency Treebank, annotated using the well-known Stanford Dependency scheme. We describe the use of the dependency treebank for PropBanking purposes and show that both annotation layers present in the treebank are highly useful for the annotation of semantic roles. We also discuss the specific features of Finnish influencing the development of a PropBank as well as the methods employed in the annotation, and finally, we present preliminary evaluation of the annotation quality.

KEYWORDS: PropBank, Finnish, dependency.

1 Introduction

Semantic role labeling (*SRL*) is one of the fundamental tasks of natural language processing. In a sense, it continues from where syntactic parsing ends: it identifies the events and participants, such as agents and patients, present in a sentence, and therefore it is an essential step in automatically processing the sentence semantics. SRL can be applied in, for example, text generation, text understanding, machine translation and fact retrieval (Palmer et al., 2005).

There have been several different efforts to capture and annotate semantic roles, the best-known projects being FrameNet (Baker et al., 1998), VerbNet (Dang et al., 1998) and PropBank (Palmer et al., 2005), all built for the English language. Out of the three resources, FrameNet is the most fine-grained one, defining roles for specific classes of verbs, such as *Cook* and *Food* for verbs relating to cooking. PropBank, in contrast, uses very generic labels, and is the only one of the three intended for corpus annotation rather than as a lexical resource. VerbNet, in turn, is between FrameNet and PropBank in granularity, and somewhat like PropBank, has close ties to syntactic structure. For a more thorough comparison of the three schemes, see the overview by Palmer et al. (2010).

The PropBank scheme in particular has become popular for semantic role labeling resources: after the initial effort on English, PropBanks for different languages have emerged, including, among others, PropBanks for Chinese (Xue and Palmer, 2009), Arabic (Zaghouani et al., 2010), Hindi (Palmer et al., 2009) and Brazilian Portuguese (Duran and Aluísio, 2011). As a PropBank is intended for corpus annotation purposes, and as the annotation scheme is closely tied to syntax, PropBanks are annotated on top of existing treebanks.

For Finnish, a freely available general language treebank has recently become available (Haverinen et al., 2010b, 2011), but no corpus annotated for semantic roles exists in the general domain. Haverinen et al. (2010a) have previously made available a small-scale PropBank of clinical Finnish, and thus shown that in principle, the PropBank scheme is suitable for Finnish and combinable with the Stanford Dependency (SD) scheme (de Marneffe and Manning, 2008a,b), the annotation scheme of both the clinical treebank and the general language treebank of Haverinen et al.

In this work, we present the first results of a project that aims to create a general language PropBank for Finnish, built on top of the existing Turku Dependency Treebank. This paper describes the methodology used for constructing the PropBank in a dependency-based manner, as well as shows the utility of the two different annotation layers present in the treebank. We also discuss the ways in which the Finnish PropBank relates to the English PropBank, our efforts to provide links between the two resources and the specific features of the Finnish language that require attention in the annotation process. Finally, we discuss the employed annotation methods and present preliminary evaluation.

2 PropBank Terminology

The purpose of a *Proposition Bank* or *PropBank*, as originally developed for English by Palmer et al. (2005), is to provide running text annotation of *semantic roles*, that is, the participants of the events described. For instance, the participants may include an *agent* who actively causes the event, or a *patient*, someone to whom the event happens. As defining a single set of roles that would cover all possible predicates is difficult, the PropBank annotation scheme defines roles on a verb-by-verb basis. Each verb receives a number of *framesets*, which can be thought of as coarse-grained senses for the verb. Each frameset consists of a *roleset*, which is a set of

act.01: to play a role, to behave		act.02: to do something	
arg0	Player	arg0	Actor
arg1	Role	arg1	Grounds for action

Figure 1: Two framesets for the verb *to act*. The frameset *act.01* is intended for usages such as *He acted as her trustee* and the frameset *act.02* for usages such as *He acted on the knowledge that she betrayed him*.

semantic roles associated with this sense of the verb, and in addition, a set of syntactic frames that describe the allowable syntactic variations.

The *roles* or *arguments* in each roleset are numbered from zero onwards. A verb can have up to six numbered arguments, although according to Palmer et al. most verbs have two to four. The arguments zero and one (Arg0 and Arg1) have specific, predefined meanings: Arg0 is reserved for *agents*, *causers* and *experiencers*, and Arg1 is used for *patients* and *themes*. The arguments Arg2 to Arg5 have no predefined meanings, but rather they are specified separately for each verb. The original PropBank project makes an effort, however, to keep also these arguments consistent within classes of verbs defined in VerbNet (Dang et al., 1998). Figure 1 illustrates two framesets for the English verb *to act*.

In addition to numbered arguments, the PropBank scheme defines so called *adjunct-like arguments* or *ArgMs*. These, unlike the numbered arguments, are not verb-specific, but rather can be applied to any verb. The original PropBank defines a set of 11 different ArgMs: *location (LOC)*, *extent (EXT)*, *discourse (DIS)*, *negation (NEG)*, *modal verb (MOD)*, *cause (CAU)*, *time (TMP)*, *purpose (PNC)*, *manner (MNR)*, *direction (DIR)* and *general purpose adverbial (ADV)*. The distinction between numbered arguments and ArgMs is made on the basis of frequency: roles that occur frequently with a particular verb sense are given numbered argument status, and less frequent roles are left as ArgMs.

PropBanks are constructed in a data-driven manner using an underlying treebank. For each different verb present in the corpus, the verb senses observed are assigned framesets in a process called *framing*, and after the framesets have been created, the occurrences in the treebank are annotated accordingly. For each verb occurrence, the annotator must select the correct frameset and mark the arguments as defined in this frameset as well as the ArgMs.

3 The Turku Dependency Treebank

This work builds on top of the previously established Turku Dependency Treebank (TDT) (Haverinen et al., 2010b, 2011), which consists of 204,399 tokens (15,126 sentences) from 10 different genres of written Finnish. The text sources of the treebank are the Finnish Wikipedia and Wikinews, popular blogs, a university online magazine, student magazines, the Finnish sections of the Europarl and JRC-Acquis corpora, a financial newspaper, grammar examples from a Finnish reference grammar and amateur fiction from various web-sources.

The syntax annotation scheme of the treebank is a Finnish-specific version of the well-known Stanford Dependency (SD) scheme (de Marneffe and Manning, 2008a,b). The SD scheme represents the syntactic structure of a sentence as a directed graph, where the nodes represent the words of the sentence and the edges represent pairwise dependencies between them. Each dependency has a direction, meaning that one of the two words connected is the *head* or *governor* and the other is the *dependent*. Each dependency also has a *type* or *label*, which describes the syntactic function of the dependent.

Figure 2: The SD scheme on a Finnish sentence. The example can be translated as *The actor has earlier lived in Italy, and moved from there to Germany*.

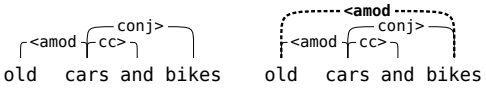

Figure 3: Conjunct propagation and coordination scope ambiguity. Left: the reading where only the cars are old. Right: The reading where both the cars and the bikes are old.

The original SD scheme contains 55 dependency types arranged in a hierarchy, where each type is a direct or indirect subtype of the most general dependency type *dependent* (*dep*). The scheme has four different *variants*, each using a different subset of the dependency types and giving a different amount of information on the sentence structure. The *basic* variant of the scheme restricts the sentence structures to trees, and the dependency types convey mostly syntactic information. The other variants add further dependencies on top of the tree structure, making the structures graphs rather than trees.

TDT uses a Finnish-specific version of the scheme, which defines a total of 53 dependency types and is described in detail in the annotation manual by Haverinen (2012). The annotation consists of two different layers of dependencies. The first annotation layer is grounded on the *basic* variant of the SD scheme, and hence the structures of the sentences in this layer are trees. The base layer of annotation is illustrated in Figure 2. The second annotation layer, termed *Conjunct propagation and additional dependencies*, adds on top of the first layer additional dependencies describing the following phenomena: *propagation of conjunct dependencies*, *external subjects* and *syntactic functions of relativizers*.

Conjunct propagation in the SD scheme provides further information on coordinations. The *basic* variant of the scheme considers the first coordinated element the head, and all other coordinated elements and the coordinating conjunction depend on it. Therefore, if a phrase modifies the first element of a coordination, it may in fact also modify all or some of the other conjuncts, and it should be *propagated* to those conjuncts that it modifies. Similarly, it is possible that all or some of the coordinated elements modify another sentence element. Conjunct propagation is used to resolve some (not all) *coordination scope ambiguities*; for instance, whether the adjective *old* modifies both *cars* and *bikes* or only *cars* in the phrase *old cars and bikes* (see Figure 3).

External subjects occur with *open clausal complements*, where a verb and its complement verb share a subject (*subject control*). The fact that the subject of the first verb is also the subject of the second verb cannot be marked in the first layer due to treeness restrictions, leaving it part of the second layer. *Relativizers*, or the phrases containing the relative word, such as *which* or *who*, are only marked as relativizers in the first layer of the treebank annotation, again in order to preserve the treeness of the structure. However, they also always have a secondary syntactic function, which in turn is annotated in the second layer of TDT. For instance, in *The man who stood at the door was tall*, the pronoun *who* acts as the subject of the verb *stood*. All of the phenomena addressed in the second layer of TDT are illustrated in Figure 4.

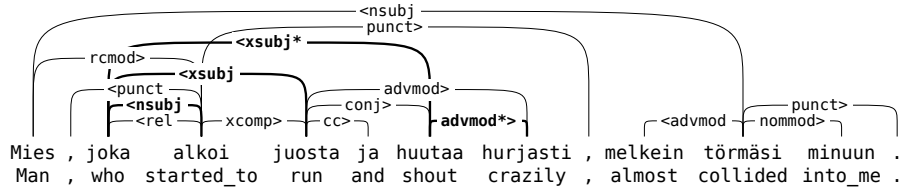

Figure 4: The second annotation layer of TDT. The example can be translated as *The man, who started to run and shout crazily, almost collided with me*. All second layer dependencies are in bold, and propagated dependencies are marked by an asterisk. The relative pronoun *joka* (*who*) also acts as the subject of the relative clause, as well an external subject to an open clausal complement. The external subject of the verb *juosta* (*run*) is also the external subject of the second coordinated verb, *huutaa* (*shout*) and is therefore propagated to the second conjunct. Similarly, the adverb modifier *hurjasti* (*crazily*) is shared between the two coordinated verbs. None of these phenomena can be accounted for in the first layer of annotation due to the treeness restriction.

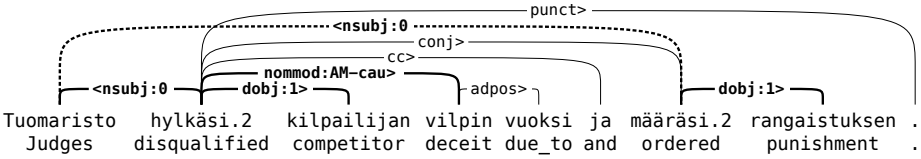

Figure 5: PropBank annotation on top of the dependency treebank. Dependencies with an associated PropBank argument are marked in bold. Note how one of the arguments (Arg0) of the latter verb in the sentence is associated with a second-layer dependency. The example sentence can be translated as *The judges disqualified the competitor due to deceit and ordered a punishment.*

4 Dependency-based PropBanking

The PropBank annotation of this work is built on top of the dependency syntax annotation of TDT, including both the first and second annotation layer. This is in contrast to the English PropBank, which has been built on top of the constituency-based Penn Treebank (Marcus et al., 1993). In the Finnish PropBank, each argument of a verb is associated with a dependency (be it first or second layer) in the underlying treebank, which means that the subtree of the dependent word, as defined by the dependencies of the first annotation layer, acts as the argument. For an illustration of the dependency-based PropBank annotation, see Figure 5.

In contrast to the original PropBank (Palmer et al., 2005) where in theory any constituent could be an argument, we make use of a heuristic: in most cases, the arguments of a verb will be its direct dependents. However, unlike the clinical language pilot study of Haverinen et al. (2010a), we do annotate all arguments, whether direct dependents of the verb or not. The heuristic of direct dependents being the likeliest arguments is only used to increase the speed of annotation by highlighting likely argument candidates for the annotator in the annotation software. In cases where an argument is found outside the dependents of the verb, we allow an extra dependency of the type *xarg* (*external argument*) to be added to any non-dependent word at annotation time, so that the argument can be attached to this dependency. For an illustration of external arguments, see Figure 6.

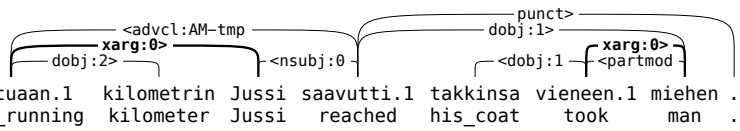

Figure 6: Arguments that are not direct dependents of the verb. On the left, the third person singular possessive suffix of the verb *juostuaan* (*after running, after he ran*) shows that it shares the subject of *saavutti* (*reached*), although this is not marked in the syntax annotation as the structure is not a case of subject control. On the right, semantically the noun *miehen* (*man*) is an argument to the verb *vieneen* (*took*), although syntactically, the verb participle modifies the noun. Note how by the assumption of whole subtrees forming arguments, the verb *vieneen* itself is incorrectly included in its own argument (Arg0) in the rightmost case. The example can be translated as *After running a kilometer, Jussi reached the man who took his coat.*

In the currently complete portion of the PropBank, 81.0% of all arguments, including both numbered arguments and ArgMs, are associated with a dependency of the first syntactic layer. If one takes into account dependencies of the second layer as well as the first, 93.1% of the arguments are covered, leaving a portion of 6.9% as external arguments. This shows that while the first layer of annotation does not suffice to cover an adequate proportion of the arguments, the second layer, which was annotated exactly for the purpose of finding semantic arguments falling outside the base-syntactic dependents of a verb, covers the majority of the remaining arguments.

As Choi and Palmer (2010) have shown, when using a dependency treebank for constructing a PropBank, in some cases the assumption that arguments are the dependents of the verb and their full subtrees results in some missing arguments that are directly due to the dependency structure of the sentence, as well as incorrect argument boundaries. In our work, the missing arguments are remedied by the *xarg* strategy, for instance in the case of a participal modifier, which is syntactically a dependent of the noun, although in fact the noun is its semantic argument. This is illustrated in Figure 6. In the case of a participal modifier, however, the addition of an *xarg* dependency leads to an incorrect argument boundary, as by the full subtree assumption the verb itself becomes part of its own argument. It should be noted that using the SD scheme already prevents some of the boundary issues mentioned by Choi and Palmer. For instance, in their work, modal verbs are problematic, as they are marked as the head of the main verb, whereas in the PropBank, the modal verb should be marked as an *ArgM-mod* for the main verb. In SD, however, the main verb is made the head and the auxiliary depends on it, which is unproblematic for PropBank annotation. A principled solution for the remaining boundary issues is not proposed in this paper, but is left as future work — perhaps using a rule-based approach, seeing that the boundary issues consist mostly of clear, regular cases.

5 Specific Features of Finnish Verbs

In the development of the Finnish PropBank, we have followed the same design principles as were used in the original PropBank: the arguments are numbered similarly from zero onwards, and the principles on which the framesets are created and distinguished are the same. We also use the same set of adjunct-like arguments, *ArgMs*, only adding two new subtypes, *consequence* (*CSQ*) and *phrasal marker* (*PRT*).

In order to expand the application potential of the Finnish PropBank to multilingual settings,

erota.2: leave a job		**quit.01: leave a job**	
Arg0	Person quitting	Arg0	Person quitting
Arg1	Job or position	Arg1	Job or position

Figure 7: Finnish and English verbs with corresponding framesets. The Finnish verb *erota* can be translated as *to quit*, and the framesets of this verb sense define identical argument structures. Therefore, the Finnish frameset is assigned the English as its corresponding frameset.

we assign to the Finnish frameset a corresponding frameset from the English PropBank where possible. Naturally, not all Finnish framesets have a corresponding English frameset, due to differences between the two languages. In this section, we discuss the specific features of the Finnish language influencing the creation of a PropBank, as well as the assignment of a corresponding English frameset and cases where no such frameset exists.

5.1 Frameset Correspondences and Non-correspondences

A frameset is assigned a corresponding English PropBank frameset when two conditions apply. The English verb must be a valid translation for the sense of the Finnish verb under consideration, and the two framesets must have the same arguments present, with matching argument numbers as well as argument descriptions. Occasionally, the argument descriptions of a corresponding English frameset are slightly rephrased in order to maximize the internal consistency of the Finnish PropBank.

As an example of corresponding framesets, one of the senses of the Finnish verb *erota* can be translated as *to quit* and it is used in contexts such as quitting a job or a position. This sense of the verb has its own frameset in the Finnish PropBank, and it is assigned a corresponding frameset in the English PropBank. The two framesets are illustrated in Figure 7.

For some verbs, however, the specific features of Finnish and the usages of the verbs being different to English do not allow assigning corresponding framesets. For instance, the frameset for the Finnish verb *korjata* meaning *to fix* or *to repair*, corresponds to neither of the English framesets, which, in turn, are also different from each other. The framesets for the three verbs are illustrated in Figure 8.

The difference between the two English framesets lies in the Arg2 argument; *to fix* includes an argument described as *benefactive*, which is absent in the description of *to repair*. The Finnish frameset, in contrast, contains an Arg2 describing an *instrument*, which is absent in both of the English framesets. Therefore it cannot be assigned either of them as the corresponding frameset. The addition of the *instrument* argument was necessary, however, as it is frequently found in the instances of the verb in the underlying treebank.

The corpus-based development of the framesets implies, naturally, that the non-correspondence of framesets does not necessarily indicate a difference between the languages. As the framesets are based on the treebank texts, they do not reflect all possible meanings and argument structures that a verb can have. This means that a non-correspondence can be caused merely by the limited and possibly different topics and text sources of the underlying treebanks. For example, the non-correspondence of the verb *korjata* with its English equivalents may be, at least partly, caused by contextual differences in the treebank texts.

A clear example of contextual differences causing non-correspondence of framesets is the Finnish

fix.02: to repair		korjata.1: to fix, to repair	
arg0	Fixer	arg0	Entity repairing something
arg1	Thing fixed	arg1	Entity repaired
arg2	Benefactive	arg2	Instrument, thing repaired with
repair.01: to restore after damage or injury			
arg0	Repairer, agent		
arg1	Entity repaired		

Figure 8: Framesets of Finnish and English verbs with the meaning *to repair*. The Finnish frameset contains an argument describing the instrument of fixing, which is not present in either of the English framesets. Note that also the two English framesets differ in that the frameset for *to fix* contains a benefactive argument, whereas the frameset for *to repair* does not.

run.02: walk quickly, course or contest		juosta.1: move rapidly on foot	
arg0	Runner	arg0	Creature running, agent
arg1	Course, race, distance	arg2	EXT, distance
arg2	Opponent	arg3	Start point
		arg4	End point

Figure 9: Framesets of Finnish and English verbs describing *running*, the rapid movement of an agent on foot. The English frameset describes running a competition or a course, as in *John ran a marathon*, and the Finnish frameset describes running from one location to another, as in *John ran from home to work*. The abbreviation *EXT* on the Finnish frameset refers to *extent*, which is one of the ArgM subtypes defined in the PropBank scheme.

verb *juosta* and its English counterpart, *to run*, both of which describe the rapid movement of an agent. In the underlying Finnish treebank, the majority of examples describe an agent running from one location to another. However, the English PropBank does not contain a frameset for such a use of the verb *to run*, but rather only a frameset describing running a competition, distance or course. This is presumably due to the Penn Treebank only containing such examples, as the English *to run* can perfectly well be used for describing movement between two locations (see for instance the Collins English dictionary (2009)). As the examples present in the Finnish treebank require a frameset whose equivalent does not exist in the English PropBank, the framesets for these two verbs are necessarily different, as illustrated in Figure 9.

5.2 Finnish Causative Verbs and Polysemous Verbs in English

In addition to verbs differing by virtue of different usages, a more systematic difference between English and Finnish verbs is caused by the verb derivation system in Finnish. In English, many verbs, especially those of movement, are polysemous and can be used in different syntactic configurations. These verbs, also termed *variable behavior verbs* (see the work of Levin and Hovav (1994) and Perlmutter (1978)), can take as their syntactic subject either an agent actively causing an event or a patient merely undergoing it. For instance, the verb *to move* can have both subject types, as in *I move* versus *The chair moves*. In addition, the verb can also be used transitively, as in *I move the chair*, where the agent causes the event undergone by the patient.

In contrast, Finnish expresses the transitive meaning using a separate verb, typically formed by adding a morpheme to the root verb. For example, from the verb *liikkua* (*to move*, intransitive), it is possible to derive *liikuttaa (to make something move)*, as illustrated in Figure 10. These *causative* verbs can be formed both from originally transitive, such as *syödä (to eat)*, the

(1)	*I move*	*Minä liikun*
(2)	*The chair moves*	*Tuoli liikkuu*
(3)	*I move the chair*	**Minä liikun tuolia*
		Minä liikutan tuolia

Figure 10: Verbs taking both agents and patients as subjects in English and in Finnish. In English, the verb *to move* has three different uses: two intransitive uses and one transitive, where the agent causes the event occurring to the patient. In Finnish, this last sense, the transitive one, is expressed by a causative verb derived from the root verb.

liikkua.1: to move, be moved		liikuttaa.1: to move something	
arg0	Entity moving actively	arg0	Entity moving arg1
arg1	Entity whose movement something causes **if not arg0**	arg1	Entity moved
arg2	Place	arg2	Place
move.01: change location			
arg0	Mover		
arg1	Moved		
arg2	Destination		

Figure 11: Framesets of Finnish and English verbs with the meaning *to move*. Top left: Finnish, intransitive verb that takes as its subject either an agent or a patient. Top right: Finnish, transitive causative verb for moving. Bottom left: English, transitive and intransitive uses. Despite appearances, the frameset *liikuttaa.1* does not correspond to the frameset *move.01*, as the English verb is allowed to take either an Arg0 or an Arg1 as its subject, whereas the Finnish verb is not.

causative being *syöttää* (*to feed*), and intransitive verbs, such as *nukkua (to sleep)*, where the causative is *nukuttaa* (*to make someone sleep*) (Hakulinen et al., 2004, §311). For causatives and causativity in general, see for instance the work of Shibatani (1976) and Itkonen (1996), and the introduction by Paulsen (2011).

For the English PropBank, all three usages of the verb *to move* can be defined by a single frameset that includes both argument zero and argument one. Depending on the arguments present in a sentence, one or both arguments can be annotated, as PropBank does not require that all arguments are present in all examples. The formulation of the Finnish framesets and the assignment of corresponding framesets is, however, more challenging.

Because of its specific argument structure, the frameset for the Finnish causative derivation *liikuttaa (to make something move)* cannot be assigned the English *to move* as its corresponding frameset; *to move* can take either an Arg0 or an Arg1 as its subject, while *liikuttaa* can not. Despite this, the verb can still have the same arguments as the English frameset. As the Finnish intransitive *liikkua (to move)* is able to take either an agent or a patient as a subject, we assign it a single frameset that contains both an Arg0 and an Arg1, and explicitly mark that these arguments are mutually exclusive, meaning that only one of them should be annotated in any given example. Figure 11 illustrates the framesets of the two Finnish verbs and for comparison, the English verb *to move*.

It is also possible, although less common, for a Finnish verb taking alternatively an agent or a patient as its subject to allow a transitive usage. An example of this is the verb *lentää* (*to fly*),

where the intransitive with an agent (*lintu lentää*, *the bird flies*), the intransitive with a patient (*lentokone lentää*, *the plane flies*) and the transitive (*pilotti lentää lentokonetta*, *the pilot flies the plane*) all use the same verb. These verbs are treated similarly to the original PropBank as they are not problematic in the same way as the verbs described above, but they are nevertheless marked as variable behavior verbs in the frameset.

6 Annotation Protocol

The annotation of the Finnish PropBank, similarly to the English one, consists of two main phases. In the first phase, each verb is given a number of framesets that describe the different senses of the verb as they occur in the underlying corpus, and in the second phase, all the occurrences of the verb are annotated according to the framesets given.

In order to recognize tokens that require PropBank annotation, we use the open source morphological analyzer OMorFi (Pirinen, 2008; Lindén et al., 2009), which gives each token all of its possible readings with no disambiguation between them. In order to ensure the annotation of all verbs in the treebank, all tokens that receive a verbal reading, or a reading indicating that the word can be a *minen*-derivation (resembles the English *ing*-participle), are selected for annotation. Calculated in this manner, the Turku Dependency Treebank contains 49,727 potential verb tokens that require annotation, and 2,946 possible different verb lemmas. At this stage, 335 lemmas have been fully annotated, resulting in a total of 9,051 annotated tokens. This means that with respect to lemmas, approximately 11.4% of the work has been completed, and with respect to tokens, the estimate is 18.2%. It should be noted that when advancing from the common verbs towards verbs with less occurrences, the annotation becomes gradually more laborious. As illustrated in Figure 12, the amount of verbs with a large amount of occurrences is fairly small as compared to the amount of verbs with only few occurrences. The framing and annotation in this project commenced not from the most common verbs but rather those with a middle range occurrence rate, in order to settle the annotation scheme before moving to the most common verbs. Thus at this stage, the verbs with the most occurrences are in fact not yet annotated.

In total six different annotators, with different amounts of previous experience and different backgrounds, contribute to the PropBank, and the same annotators also act as framers. The verbs present in the treebank are framed and annotated one lemma at a time. In the beginning of the annotation process, all occurrences of each lemma were double annotated, in order to ensure high annotation quality even in the beginning phases of the project. As the work has progressed, we have gradually moved towards single annotation; high-frequency lemmas are partially double annotated, while low-frequency lemmas are single annotated. This is to increase the annotation speed while still being able to measure and control the annotation quality even after the initial learning phase.

In the case of double annotation, the two annotators assigned to a lemma create the framesets jointly, after which both of them independently annotate all occurrences using these framesets. At this stage, the annotator is required to mark both the numbered arguments and the adjunct-like arguments present in each example. Afterwards, the two analyses of each example are automatically merged, so that all disagreements can easily be seen, and in a meeting between all annotators, a single correct analysis is decided upon. Partially double annotated lemmas are framed in co-operation, and a portion of the occurrences is double annotated while the rest are divided between the annotators. In single annotation, each lemma is given to one annotator, and additionally, one annotator is assigned as a *consultant*, whom the annotator of the lemma

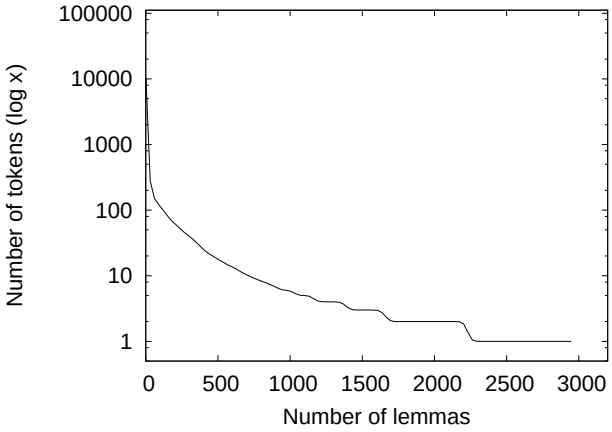

Figure 12: Numbers of verb lemmas of different frequencies as sorted from the highest number of occurrences to the lowest. High-frequency lemmas are relatively few, while many different low-frequency lemmas occur in the treebank text.

can turn to if facing problems with the framing. If unsure in the annotation phase, be it double or single annotation, an annotator can mark any argument as unsure. This function can also be used to signal suspected syntax-level errors in the treebank, as annotators are not allowed to alter the syntax at this stage.

In order to alleviate the labor-intensity of creating the framesets, *batches* of similar verbs are given their framesets simultaneously. When creating a new frameset for a lemma, the annotator is to consider whether there are other verbs that should also receive the same frameset, if such verbs are easily found. (The opposite is also possible: when considering a lemma, if the annotator finds that an existing frameset from another lemma can be re-used, they may copy the desired frameset for the verb under consideration.) For instance, if an annotator is considering the verb *to like*, possible other verbs that could receive the same frameset would be *to love*, *to care* or other verbs expressing affection that may have the same arguments. However, simply having the same arguments as in numbered arguments is not sufficient to be included in the same batch: for instance, verbs of dislike, although they also receive arguments describing the experiencer and the object of the feeling, should not be assigned to the same batch as the verbs of affection. In order to be included in the same batch, the verbs must have the same numbered arguments, and also the argument descriptions are required to be suitable for all verbs included.

This strategy has two benefits: in addition to saving time by creating framesets practically with no additional cost, it can enforce some consistency across the verbs. As a minor drawback, it requires additional care, as annotators should always make sure that the lemma they are considering does not already have the intended frameset as a side product of some other lemma. Also, if making a frameset for several verbs at once, care should be taken that verbs assigned simultaneously to other annotators do not receive framesets without these annotators' knowledge.

The distinctions between different framesets are made according to guidelines similar to those

used in the English PropBank, that is, the verb senses that the framesets correspond to are fairly coarse-grained. The main criterion used is that if two potential framesets have the same arguments (including the descriptions), or the arguments of one are a subset of the arguments of the other, only one frameset should be created.

The annotation is done using a custom software (see Figure 13) that allows the annotator to select a lemma to be annotated and then displays each occurrence as a separate case. The annotator must first select the correct frameset for the occurrence under consideration, and then assign the numbered arguments and adjunct-like arguments. All dependents of the verb occurrence are highlighted as default options for arguments, except for certain dependency types, such as punctuation, which never act as arguments. In case a dependency does not correspond to an argument, it is possible to leave the dependency unmarked. In addition, it is possible to mark a sentence element not depending on the verb as an argument using the *external argument* dependency. In addition to choosing one of the framesets defined, it is also possible to take one of the following actions. First, the annotator can mark an occurrence as *not a verb*, where the token is not in fact a verb but rather another part-of-speech, despite having a verbal reading assigned by OMorFi. Second, similarly it is possible mark the token to have a *wrong lemma*, where the token is a verb, but not of the lemma currently under consideration. Third, it is possible to mark the occurrence as an *auxiliary*, as in the PropBank scheme auxiliaries do not receive framesets or arguments.

7 Evaluation

In order to evaluate the performance of the annotators, we measure their *annotator accuracy* against the *merged* annotation. The accuracy is calculated using F_1-score, which is defined as $F_1 = \frac{2PR}{P+R}$. *Precision* (P) is the percentage of arguments in the individual annotation that are also present in the merged annotation, and *recall* (R) the percentage of arguments in the merged annotation that are also present in the individual annotation. For an argument to be considered correct, both its dependent word (the head word is the verb and thus always correct) and the argument number or the ArgM type must be correct. If the frameset of a verb is incorrect, then all numbered arguments of this verb token are considered incorrect as well. An ArgM of the correct type is judged correct regardless of the frameset of the verb, as ArgMs are verb-independent. For comparison, we also calculate *inter-annotator agreement* using *Cohen's kappa*, defined as $\kappa = \frac{P(A)-P(E)}{1-P(E)}$, where *P(A)* is the observed agreement and *P(E)* is the agreement expected by chance.

The overall annotator accuracy on the whole task is 94.1%, and the overall inter-annotator agreement in Kappa is 89.7%. While the F_1-score measures the accuracy of an annotator against the merged gold-standard, the Kappa-score measures the agreement between the annotators. It should also be noted that as the Kappa-score can only reasonably be calculated for labeling tasks, the external arguments, that is, arguments that are not syntactically direct dependents of the verb, are only taken into account in the F_1-score and not in Kappa. The per-annotator accuracies in F_1-score are listed in Table 1. The Table lists both overall scores and scores on numbered arguments and adjunct-like arguments separately, as well as the external arguments. These results show that overall, the accuracy is high, and that the adjunct-like arguments are more difficult to annotate than the numbered arguments, which is an expected result based on the figures previously reported by Palmer et al. (2005). The external arguments also seem to be more difficult than the numbered arguments in general. Some annotators show a large difference between precision and recall on the external arguments, indicating that these

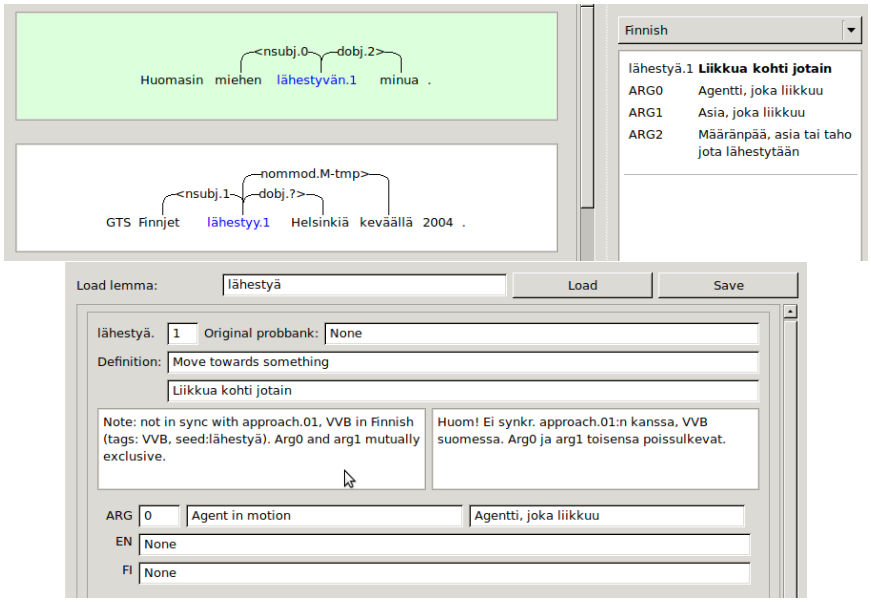

Figure 13: The annotation tool. Top: occurrence annotation. Two occurrences to be annotated are shown on the left, one marked as ready and one in mid-annotation. The direct dependents of the verb are shown as default alternatives for the arguments, and the question mark in the latter example indicates a dependency that has not been assigned with an argument. The framesets that have been created for the verb currently being annotated are shown on the right. Bottom: frameset editor. Each frameset has a number, a description, a field for the corresponding English PropBank frameset (not set in this example), as well as a free comment field. Similarly, each argument has a number, a description and a comment field. The comment fields may be used, for instance, for case requirements or use examples.

	Ann. 1	Ann. 2	Ann. 3	Ann. 4	Ann. 5	Ann. 6	All
Numbered (n=29,076)							
Recall	98.1	96.5	96.5	94.9	97.8	95.6	96.9
Precision	98.5	98.0	98.0	95.1	98.1	94.5	97.4
F-score	98.3	97.2	97.3	95.0	97.9	95.1	97.1
ArgM (n=15,771)							
Recall	92.5	86.6	87.3	83.7	90.1	82.8	87.8
Precision	92.9	87.3	86.6	85.2	92.6	82.0	88.2
F-score	92.7	86.9	87.0	84.4	91.3	82.4	88.0
xarg (n=3,118)							
Recall	93.3	80.8	79.3	70.3	87.4	85.9	86.0
Precision	97.8	97.8	92.3	70.3	94.7	84.3	92.7
F-score	95.5	88.5	85.3	70.3	90.9	85.1	89.2
overall (n=44,847)							
Recall	96.3	93.0	93.4	90.9	95.2	91.6	93.9
Precision	96.7	94.2	94.1	91.5	96.3	90.6	94.3
F-score	96.5	93.6	93.7	91.2	95.8	91.1	94.1

Table 1: Annotator accuracy results per annotator, both overall and separately for numbered arguments and ArgMs. Also a separate evaluation of the external arguments (*xarg*) is given. Note that for the F_1-scores the external arguments are also included in the counts of numbered arguments and ArgMs, seeing that each external argument is also one of these two argument types.

annotators forget to mark an external argument more often than mark an extraneous one. In addition to the possibility of overlooking an external argument, the task is made more difficult by the fact that with *xargs*, unlike the other arguments, the annotator is required to identify the correct token to act as the dependent.

Further, we evaluate the correctness of the frameset selections. Out of all frameset choices (including the possible choices of *not a verb*, *wrong lemma* and *auxiliary*), 88.4% were correct as measured against the final annotation result. Measured on only those instances where the frameset was correctly selected, the overall F_1-score was 94.6%.

8 Conclusions

In this work, we have presented the first results from a project aiming at a general Finnish PropBank. This PropBank is built on top of the previously existing Turku Dependency Treebank and utilizes both the first and second layers of syntax annotation present in the treebank, which are annotated according to the Stanford Dependency scheme.

We confirm the preliminary finding of the clinical language pilot study by Haverinen et al. (2010a) that the PropBank scheme can be used for Finnish and is compatible with the SD scheme. We also find that a large number of arguments are covered by the simplifying assumption that arguments are syntactic dependents of the verb; 81.0% of all arguments are accounted for when only considering the first layer of syntax annotation in TDT, and 93.1% if also the second layer is taken into consideration.

Regarding the quality of annotation, we find that the overall annotator accuracy of all six different annotators is 94.1% in F_1-score, and the accuracy on adjunct-like arguments (ArgMs) alone is 88.0%. The inter-annotator agreement in Cohen's kappa on the overall task disregarding

external arguments is 89.7%. From these figures we conclude that overall the quality of annotation is high, and that as expected, the adjunct-like arguments are more difficult to annotate than the numbered arguments. External arguments, with an overall F_1-score of 89.2%, are also more difficult than numbered arguments in general, due to the possibility of overlooking an external argument as well as the fact that for these arguments, the annotator also needs to identify the correct dependent word.

As future work, in addition to increasing the coverage of the PropBank, it would be beneficial to build rules to treat cases where the full subtree assumption of arguments fails, as well as enhance the annotation towards noun argument structures, that is, a NomBank (Meyers et al., 2004). The annotation could also be enhanced in several ways in order to accommodate, for instance, text generation, along the guidelines suggested by Wanner et al. (2012).

Acknowledgments

This work has been supported by the Academy of Finland and the Emil Aaltonen Foundation.

References

(2009). *Collins English Dictionary — 30th Anniversary Edition*. HarperCollins Publishers.

Baker, C. F., Fillmore, C. J., and Lowe, J. B. (1998). The Berkeley FrameNet project. In *Proceedings of COLING-ACL'98*, pages 86–90.

Choi, J. and Palmer, M. (2010). Retrieving correct semantic boundaries in dependency structure. In *Proceedings of LAW IV*, pages 91–99.

Dang, H. T., Kipper, K., Palmer, M., and Rosenzweig, J. (1998). Investigating regular sense extensions based on intersective Levin classes. In *Proceedings of COLING-ACL'98*, pages 293–299.

Duran, M. S. and Aluísio, S. M. (2011). Propbank-br: a Brazilian treebank annotated with semantic role labels. In *Proceedings of STIL'11*, pages 1862–1867.

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

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

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

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

Haverinen, K., Viljanen, T., Laippala, V., Kohonen, S., Ginter, F., and Salakoski, T. (2010b). Treebanking Finnish. In Dickinson, M., Müürisep, K., and Passarotti, M., editors, *Proceedings of The ninth International Workshop on Treebanks and Linguistic Theories (TLT9)*, pages 79–90.

Itkonen, E. (1996). *Maailman kielten erilaisuus ja samuus / Differences and Similarities of the World Languages*. Gaudeamus.

Levin, B. and Hovav, M. R. (1994). *Unaccusativity: At the syntax–lexical semantics interface*, volume 26 of *Linguistic Inquiry*. MIT Press.

Lindén, K., Silfverberg, M., and 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*, volume 41 of *Communications in Computer and Information Science*, pages 28–47.

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

de Marneffe, M.-C. and Manning, C. (2008a). Stanford typed dependencies manual. Technical report, Stanford University.

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

Meyers, A., Reeves, R., Macleod, C., Szekely, R., Zielinska, V., Young, B., and Grishman, R. (2004). The NomBank project: An interim report. In *In Proceedings of the NAACL/HLT Workshop on Frontiers in Corpus Annotation*.

Palmer, M., Bhatt, R., Narasimhan, B., Rambow, O., Sharma, D. M., and Xia, F. (2009). Hindi syntax: annotating dependency, lexical predicate–argument structure, and phrase structure. In *Proceedings of ICON'09*.

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

Palmer, M., Gildea, D., and Xue, N. (2010). *Semantic Role Labeling*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Paulsen, G. (2011). *Causation and Dominance*. PhD thesis.

Perlmutter, D. (1978). Impersonal passives and the unaccusative hypothesis. In *Proceedings of the Fourth Annual Meeting of the Berkeley Linguistic Society*, pages 157–189.

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

Shibatani, M., editor (1976). *The Grammar of Causative Constructions*, volume 6 of *Syntax and Semantics*. Seminar Press.

Wanner, L., Mille, S., and Bohnet, B. (2012). Towards a surface realization-oriented corpus annotation. In *Proceedings of INLG '12*, pages 22–30.

Xue, N. and Palmer, M. (2009). Adding semantic roles to the Chinese treebank. *Natural Language Engineering*, 15(Special issue 01):143–172.

Zaghouani, W., Diab, M., Mansouri, A., Pradhan, S., and Palmer, M. (2010). The revised Arabic PropBank. In *Proceedings of LAW IV*, pages 222–226.

Turku Centre for Computer Science
TUCS Dissertations

1. **Marjo Lipponen**, On Primitive Solutions of the Post Correspondence Problem
2. **Timo Käkölä**, Dual Information Systems in Hyperknowledge Organizations
3. **Ville Leppänen**, Studies on the Realization of PRAM
4. **Cunsheng Ding**, Cryptographic Counter Generators
5. **Sami Viitanen**, Some New Global Optimization Algorithms
6. **Tapio Salakoski**, Representative Classification of Protein Structures
7. **Thomas Långbacka**, An Interactive Environment Supporting the Development of Formally Correct Programs
8. **Thomas Finne**, A Decision Support System for Improving Information Security
9. **Valeria Mihalache**, Cooperation, Communication, Control. Investigations on Grammar Systems.
10. **Marina Waldén**, Formal Reasoning About Distributed Algorithms
11. **Tero Laihonen**, Estimates on the Covering Radius When the Dual Distance is Known
12. **Lucian Ilie**, Decision Problems on Orders of Words
13. **Jukkapekka Hekanaho**, An Evolutionary Approach to Concept Learning
14. **Jouni Järvinen**, Knowledge Representation and Rough Sets
15. **Tomi Pasanen**, In-Place Algorithms for Sorting Problems
16. **Mika Johnsson**, Operational and Tactical Level Optimization in Printed Circuit Board Assembly
17. **Mats Aspnäs**, Multiprocessor Architecture and Programming: The Hathi-2 System
18. **Anna Mikhajlova**, Ensuring Correctness of Object and Component Systems
19. **Vesa Torvinen**, Construction and Evaluation of the Labour Game Method
20. **Jorma Boberg**, Cluster Analysis. A Mathematical Approach with Applications to Protein Structures
21. **Leonid Mikhajlov**, Software Reuse Mechanisms and Techniques: Safety Versus Flexibility
22. **Timo Kaukoranta**, Iterative and Hierarchical Methods for Codebook Generation in Vector Quantization
23. **Gábor Magyar**, On Solution Approaches for Some Industrially Motivated Combinatorial Optimization Problems
24. **Linas Laibinis**, Mechanised Formal Reasoning About Modular Programs
25. **Shuhua Liu**, Improving Executive Support in Strategic Scanning with Software Agent Systems
26. **Jaakko Järvi**, New Techniques in Generic Programming – C++ is more Intentional than Intended
27. **Jan-Christian Lehtinen**, Reproducing Kernel Splines in the Analysis of Medical Data
28. **Martin Büchi**, Safe Language Mechanisms for Modularization and Concurrency
29. **Elena Troubitsyna**, Stepwise Development of Dependable Systems
30. **Janne Näppi**, Computer-Assisted Diagnosis of Breast Calcifications
31. **Jianming Liang**, Dynamic Chest Images Analysis
32. **Tiberiu Seceleanu**, Systematic Design of Synchronous Digital Circuits
33. **Tero Aittokallio**, Characterization and Modelling of the Cardiorespiratory System in Sleep-Disordered Breathing
34. **Ivan Porres**, Modeling and Analyzing Software Behavior in UML
35. **Mauno Rönkkö**, Stepwise Development of Hybrid Systems
36. **Jouni Smed**, Production Planning in Printed Circuit Board Assembly
37. **Vesa Halava**, The Post Correspondence Problem for Market Morphisms
38. **Ion Petre**, Commutation Problems on Sets of Words and Formal Power Series
39. **Vladimir Kvassov**, Information Technology and the Productivity of Managerial Work
40. **Frank Tétard**, Managers, Fragmentation of Working Time, and Information Systems

41. **Jan Manuch**, Defect Theorems and Infinite Words
42. **Kalle Ranto**, Z_4-Goethals Codes, Decoding and Designs
43. **Arto Lepistö**, On Relations Between Local and Global Periodicity
44. **Mika Hirvensalo**, Studies on Boolean Functions Related to Quantum Computing
45. **Pentti Virtanen**, Measuring and Improving Component-Based Software Development
46. **Adekunle Okunoye**, Knowledge Management and Global Diversity – A Framework to Support Organisations in Developing Countries
47. **Antonina Kloptchenko**, Text Mining Based on the Prototype Matching Method
48. **Juha Kivijärvi**, Optimization Methods for Clustering
49. **Rimvydas Rukšėnas**, Formal Development of Concurrent Components
50. **Dirk Nowotka**, Periodicity and Unbordered Factors of Words
51. **Attila Gyenesei**, Discovering Frequent Fuzzy Patterns in Relations of Quantitative Attributes
52. **Petteri Kaitovaara**, Packaging of IT Services – Conceptual and Empirical Studies
53. **Petri Rosendahl**, Niho Type Cross-Correlation Functions and Related Equations
54. **Péter Majlender**, A Normative Approach to Possibility Theory and Soft Decision Support
55. **Seppo Virtanen**, A Framework for Rapid Design and Evaluation of Protocol Processors
56. **Tomas Eklund**, The Self-Organizing Map in Financial Benchmarking
57. **Mikael Collan**, Giga-Investments: Modelling the Valuation of Very Large Industrial Real Investments
58. **Dag Björklund**, A Kernel Language for Unified Code Synthesis
59. **Shengnan Han**, Understanding User Adoption of Mobile Technology: Focusing on Physicians in Finland
60. **Irina Georgescu**, Rational Choice and Revealed Preference: A Fuzzy Approach
61. **Ping Yan**, Limit Cycles for Generalized Liénard-Type and Lotka-Volterra Systems
62. **Joonas Lehtinen**, Coding of Wavelet-Transformed Images
63. **Tommi Meskanen,** On the NTRU Cryptosystem
64. **Saeed Salehi**, Varieties of Tree Languages
65. **Jukka Arvo**, Efficient Algorithms for Hardware-Accelerated Shadow Computation
66. **Mika Hirvikorpi**, On the Tactical Level Production Planning in Flexible Manufacturing Systems
67. **Adrian Costea**, Computational Intelligence Methods for Quantitative Data Mining
68. **Cristina Seceleanu**, A Methodology for Constructing Correct Reactive Systems
69. **Luigia Petre**, Modeling with Action Systems
70. **Lu Yan**, Systematic Design of Ubiquitous Systems
71. **Mehran Gomari**, On the Generalization Ability of Bayesian Neural Networks
72. **Ville Harkke**, Knowledge Freedom for Medical Professionals – An Evaluation Study of a Mobile Information System for Physicians in Finland
73. **Marius Cosmin Codrea**, Pattern Analysis of Chlorophyll Fluorescence Signals
74. **Aiying Rong**, Cogeneration Planning Under the Deregulated Power Market and Emissions Trading Scheme
75. **Chihab BenMoussa**, Supporting the Sales Force through Mobile Information and Communication Technologies: Focusing on the Pharmaceutical Sales Force
76. **Jussi Salmi**, Improving Data Analysis in Proteomics
77. **Orieta Celiku**, Mechanized Reasoning for Dually-Nondeterministic and Probabilistic Programs
78. **Kaj-Mikael Björk**, Supply Chain Efficiency with Some Forest Industry Improvements
79. **Viorel Preoteasa**, Program Variables – The Core of Mechanical Reasoning about Imperative Programs
80. **Jonne Poikonen**, Absolute Value Extraction and Order Statistic Filtering for a Mixed-Mode Array Image Processor
81. **Luka Milovanov**, Agile Software Development in an Academic Environment
82. **Francisco Augusto Alcaraz Garcia**, Real Options, Default Risk and Soft Applications
83. **Kai K. Kimppa**, Problems with the Justification of Intellectual Property Rights in Relation to Software and Other Digitally Distributable Media
84. **Dragoş Truşcan**, Model Driven Development of Programmable Architectures
85. **Eugen Czeizler**, The Inverse Neighborhood Problem and Applications of Welch Sets in Automata Theory

86. **Sanna Ranto**, Identifying and Locating-Dominating Codes in Binary Hamming Spaces
87. **Tuomas Hakkarainen**, On the Computation of the Class Numbers of Real Abelian Fields
88. **Elena Czeizler**, Intricacies of Word Equations
89. **Marcus Alanen**, A Metamodeling Framework for Software Engineering
90. **Filip Ginter**, Towards Information Extraction in the Biomedical Domain: Methods and Resources
91. **Jarkko Paavola**, Signature Ensembles and Receiver Structures for Oversaturated Synchronous DS-CDMA Systems
92. **Arho Virkki**, The Human Respiratory System: Modelling, Analysis and Control
93. **Olli Luoma**, Efficient Methods for Storing and Querying XML Data with Relational Databases
94. **Dubravka Ilić**, Formal Reasoning about Dependability in Model-Driven Development
95. **Kim Solin**, Abstract Algebra of Program Refinement
96. **Tomi Westerlund**, Time Aware Modelling and Analysis of Systems-on-Chip
97. **Kalle Saari**, On the Frequency and Periodicity of Infinite Words
98. **Tomi Kärki**, Similarity Relations on Words: Relational Codes and Periods
99. **Markus M. Mäkelä**, Essays on Software Product Development: A Strategic Management Viewpoint
100. **Roope Vehkalahti**, Class Field Theoretic Methods in the Design of Lattice Signal Constellations
101. **Anne-Maria Ernvall-Hytönen**, On Short Exponential Sums Involving Fourier Coefficients of Holomorphic Cusp Forms
102. **Chang Li**, Parallelism and Complexity in Gene Assembly
103. **Tapio Pahikkala**, New Kernel Functions and Learning Methods for Text and Data Mining
104. **Denis Shestakov**, Search Interfaces on the Web: Querying and Characterizing
105. **Sampo Pyysalo**, A Dependency Parsing Approach to Biomedical Text Mining
106. **Anna Sell**, Mobile Digital Calendars in Knowledge Work
107. **Dorina Marghescu**, Evaluating Multidimensional Visualization Techniques in Data Mining Tasks
108. **Tero Säntti**, A Co-Processor Approach for Efficient Java Execution in Embedded Systems
109. **Kari Salonen**, Setup Optimization in High-Mix Surface Mount PCB Assembly
110. **Pontus Boström**, Formal Design and Verification of Systems Using Domain-Specific Languages
111. **Camilla J. Hollanti**, Order-Theoretic Mehtods for Space-Time Coding: Symmetric and Asymmetric Designs
112. **Heidi Himmanen**, On Transmission System Design for Wireless Broadcasting
113. **Sébastien Lafond**, Simulation of Embedded Systems for Energy Consumption Estimation
114. **Evgeni Tsivtsivadze**, Learning Preferences with Kernel-Based Methods
115. **Petri Salmela**, On Commutation and Conjugacy of Rational Languages and the Fixed Point Method
116. **Siamak Taati**, Conservation Laws in Cellular Automata
117. **Vladimir Rogojin**, Gene Assembly in Stichotrichous Ciliates: Elementary Operations, Parallelism and Computation
118. **Alexey Dudkov**, Chip and Signature Interleaving in DS CDMA Systems
119. **Janne Savela**, Role of Selected Spectral Attributes in the Perception of Synthetic Vowels
120. **Kristian Nybom**, Low-Density Parity-Check Codes for Wireless Datacast Networks
121. **Johanna Tuominen**, Formal Power Analysis of Systems-on-Chip
122. **Teijo Lehtonen**, On Fault Tolerance Methods for Networks-on-Chip
123. **Eeva Suvitie**, On Inner Products Involving Holomorphic Cusp Forms and Maass Forms
124. **Linda Mannila**, Teaching Mathematics and Programming – New Approaches with Empirical Evaluation
125. **Hanna Suominen**, Machine Learning and Clinical Text: Supporting Health Information Flow
126. **Tuomo Saarni**, Segmental Durations of Speech
127. **Johannes Eriksson**, Tool-Supported Invariant-Based Programming

128. **Tero Jokela**, Design and Analysis of Forward Error Control Coding and Signaling for Guaranteeing QoS in Wireless Broadcast Systems
129. **Ville Lukkarila**, On Undecidable Dynamical Properties of Reversible One-Dimensional Cellular Automata
130. **Qaisar Ahmad Malik**, Combining Model-Based Testing and Stepwise Formal Development
131. **Mikko-Jussi Laakso**, Promoting Programming Learning: Engagement, Automatic Assessment with Immediate Feedback in Visualizations
132. **Riikka Vuokko**, A Practice Perspective on Organizational Implementation of Information Technology
133. **Jeanette Heidenberg**, Towards Increased Productivity and Quality in Software Development Using Agile, Lean and Collaborative Approaches
134. **Yong Liu**, Solving the Puzzle of Mobile Learning Adoption
135. **Stina Ojala**, Towards an Integrative Information Society: Studies on Individuality in Speech and Sign
136. **Matteo Brunelli**, Some Advances in Mathematical Models for Preference Relations
137. **Ville Junnila**, On Identifying and Locating-Dominating Codes
138. **Andrzej Mizera**, Methods for Construction and Analysis of Computational Models in Systems Biology. Applications to the Modelling of the Heat Shock Response and the Self-Assembly of Intermediate Filaments.
139. **Csaba Ráduly-Baka**, Algorithmic Solutions for Combinatorial Problems in Resource Management of Manufacturing Environments
140. **Jari Kyngäs**, Solving Challenging Real-World Scheduling Problems
141. **Arho Suominen**, Notes on Emerging Technologies
142. **József Mezei**, A Quantitative View on Fuzzy Numbers
143. **Marta Olszewska**, On the Impact of Rigorous Approaches on the Quality of Development
144. **Antti Airola**, Kernel-Based Ranking: Methods for Learning and Performace Estimation
145. **Aleksi Saarela**, Word Equations and Related Topics: Independence, Decidability and Characterizations
146. **Lasse Bergroth**, Kahden merkkijonon pisimmän yhteisen alijonon ongelma ja sen ratkaiseminen
147. **Thomas Canhao Xu**, Hardware/Software Co-Design for Multicore Architectures
148. **Tuomas Mäkilä**, Software Development Process Modeling – Developers Perspective to Contemporary Modeling Techniques
149. **Shahrokh Nikou**, Opening the Black-Box of IT Artifacts: Looking into Mobile Service Characteristics and Individual Perception
150. **Alessandro Buoni**, Fraud Detection in the Banking Sector: A Multi-Agent Approach
151. **Mats Neovius**, Trustworthy Context Dependency in Ubiquitous Systems
152. **Fredrik Degerlund**, Scheduling of Guarded Command Based Models
153. **Amir-Mohammad Rahmani-Sane**, Exploration and Design of Power-Efficient Networked Many-Core Systems
154. **Ville Rantala**, On Dynamic Monitoring Methods for Networks-on-Chip
155. **Mikko Pelto**, On Identifying and Locating-Dominating Codes in the Infinite King Grid
156. **Anton Tarasyuk**, Formal Development and Quantitative Verification of Dependable Systems
157. **Muhammad Mohsin Saleemi**, Towards Combining Interactive Mobile TV and Smart Spaces: Architectures, Tools and Application Development
158. **Tommi J. M. Lehtinen**, Numbers and Languages
159. **Peter Sarlin**, Mapping Financial Stability
160. **Alexander Wei Yin**, On Energy Efficient Computing Platforms
161. **Mikołaj Olszewski**, Scaling Up Stepwise Feature Introduction to Construction of Large Software Systems
162. **Maryam Kamali**, Reusable Formal Architectures for Networked Systems
163. **Zhiyuan Yao**, Visual Customer Segmentation and Behavior Analysis – A SOM-Based Approach
164. **Timo Jolivet**, Combinatorics of Pisot Substitutions
165. **Rajeev Kumar Kanth**, Analysis and Life Cycle Assessment of Printed Antennas for Sustainable Wireless Systems
166. **Khalid Latif**, Design Space Exploration for MPSoC Architectures

167. **Bo Yang**, Towards Optimal Application Mapping for Energy-Efficient Many-Core Platforms
168. **Ali Hanzala Khan**, Consistency of UML Based Designs Using Ontology Reasoners
169. **Sonja Leskinen**, m-Equine: IS Support for the Horse Industry
170. **Fareed Ahmed Jokhio**, Video Transcoding in a Distributed Cloud Computing Environment
171. **Moazzam Fareed Niazi**, A Model-Based Development and Verification Framework for Distributed System-on-Chip Architecture
172. **Mari Huova**, Combinatorics on Words: New Aspects on Avoidability, Defect Effect, Equations and Palindromes
173. **Ville Timonen**, Scalable Algorithms for Height Field Illumination
174. **Henri Korvela**, Virtual Communities – A Virtual Treasure Trove for End-User Developers
175. **Kameswar Rao Vaddina**, Thermal-Aware Networked Many-Core Systems
176. **Janne Lahtiranta**, New and Emerging Challenges of the ICT-Mediated Health and Well-Being Services
177. **Irum Rauf**, Design and Validation of Stateful Composite RESTful Web Services
178. **Jari Björne**, Biomedical Event Extraction with Machine Learning
179. **Katri Haverinen**, Natural Language Processing Resources for Finnish: Corpus Development in the General and Clinical Domains

Turku Centre *for* Computer Science

Joukahaisenkatu 3-5 B, 20520 Turku, Finland | www. tucs.fi

University of Turku
Faculty of Mathematics and Natural Sciences
- Department of Information Technology
- Department of Mathematics and Statistics

Turku School of Economics
- Institute of Information Systems Science

Åbo Akademi University
Division for Natural Sciences and Technology
- Department of Information Technologies

ISBN 978-952-12-3083-7
ISSN 1239-1883

Katri Haverinen

Natural Language Processing Resources for Finnish

Katri Haverinen

Natural Language Processing Resources for Finnish