Predicting conjunct propagation and other extended Stanford Dependencies

Jenna Nyblom[1], **Samuel Kohonen**[1], **Katri Haverinen**[1,2], **Tapio Salakoski**[1,2] and **Filip Ginter**[1]
[1]Department of Information Technology, University of Turku, Finland
[2]Turku Centre for Computer Science, Turku, Finland
first.last@utu.fi

Abstract

In this work, we present a data-driven method to enhance syntax trees with additional dependencies as defined in the well-known Stanford Dependencies scheme, so as to give more information about the structure of the sentence. This hybrid method utilizes both machine learning and a rule-based approach, and achieves a performance of 93.1% in F_1-score, as evaluated using an existing treebank of Finnish. The resulting tool will be integrated into an existing Finnish parser and made publicly available at the address http://bionlp.utu.fi/.

1 Introduction

Dependency-based analysis of syntax has recently become popular within natural language processing. It has been argued to be preferable over constituency analysis in both parser evaluation and further applications (Lin, 1998; Clegg and Shepherd, 2007), and indeed both dependency treebanks and parsers have emerged in recent years.

Dependency formalisms usually require that all valid analyses must be *trees*, meaning that each token in a sentence must only have one governor, and the whole sentence must have one head word. Tree structures, however, do not necessarily allow the explicit representation of a number of relevant phenomena. This is demonstrated by the well-known Stanford Dependencies (SD) scheme (de Marneffe and Manning, 2008), which is defined in multiple *variants*. The *basic* variant requires sentence structures to be trees, and the other variants can then be used to add further dependencies on top of the tree structure, making the resulting structures graphs rather than trees. Phenomena that are further analyzed in the non-basic variants of SD include relative clauses, open clausal complements, coordinations and prepositional phrases.

The dependencies present in non-basic variants of SD can be useful for applications that build on top of the syntactic analysis. For instance, the clinical domain pilot study of Haverinen et al. (2010) has shown that these dependencies can be used in annotating argument structures of verbs using the popular PropBank scheme (Palmer et al., 2005). Also, Yuret et al. (2012) have used the propagated and collapsed variant of the SD scheme to retrieve as semantically meaningful dependencies as possible in the context of textual entailments. The non-basic variants of SD are also extensively applied in information extraction, as seen for example in the BioNLP shared tasks on event extraction, where a number of top-ranking systems relied on SD analyses (Kim et al., 2011).

In this work, we are concerned with three phenomena represented in the non-basic variants of SD. Most importantly, we consider the dependencies that are the result of *conjunct propagation*. They resolve, at least partially, ambiguities known as *coordination scope ambiguities*. These are ambiguities where there are multiple ways to understand the scope of a coordination; for instance, in the phrase *old men and women* either both the men and the women are old, or alternatively, only the men. Additionally, we consider dependencies that reveal the *syntactic functions of relativizers* and *external subjects* of open clausal complements.

We present a method that, given the basic syntactic tree of a sentence, predicts these additional dependencies as defined in the SD scheme using machine learning. As training data, we use morphological and syntactic information gathered from an existing treebank of Finnish, which has human annotated conjunct propagation and additional dependencies present. We begin with a discussion of related work and the treebank used as training material. We then move on to the details of the method itself and present a thorough evaluation of the pipeline. We make comparisons with

Proceedings of the Second International Conference on Dependency Linguistics (DepLing 2013), pages 252–261,
Prague, August 27–30, 2013.

several baseline methods, and conclude that the proposed method achieves a performance clearly superior to each of these baselines. In particular, the method demonstrates performance clearly superior to that achieved by the commonly used Stanford tools.

2 Related work

The general problem of *coordination scope ambiguity* is a widely studied, difficult problem. It is frequently tackled by utilizing lexical parallelism and selectional preferences, as for instance in the works of Kawahara and Kurohashi (2011) and Resnik (1999). In the domain of requirements engineering, Chantree et al. (2005) disambiguate coordinations using heuristics based on the distributions of the words appearing in them. Goldberg (1999) has presented an unsupervised model for a limited range of coordination phenomena, and Agarwal and Boggess (1992) introduce a simplified algorithm for recognizing the correct conjuncts for coordinations. Kawahara and Kurohashi (2007) and van Noord (2007) have incorporated disambiguation methods into parsers of Japanese and Dutch, respectively.

In dependency representations, there are multiple ways to treat coordination structures, and the chosen treatment also affects coordination scope ambiguities. The Stanford Dependencies scheme (de Marneffe and Manning, 2008) used in this work considers the first coordinated element the head of the coordination, and uses an additional layer of dependencies to represent the propagation of conjunct dependencies (see Figures 1 and 2). As a point of comparison, for instance the Link Grammar scheme (Sleator and Temperley, 1993) makes the coordinating conjunction the head word of the coordination, thus partially resolving the scope ambiguities using tree structures only as will be shown in greater detail in Section 5.2.

The Stanford tools[1] are able to produce output with the additional dependencies of the SD scheme present, but according to de Marneffe and Manning (2008), this part of the tools performs imperfectly. While the English resource PARC 700 (King et al., 2003), annotated in the LFG-formalism (Bresnan, 2001), contains dependencies similar to those considered in this work, to our knowledge the Turku Dependency Treebank (Haverinen et al., 2011) is the only existing manually annotated resource that contains conjunct propagation as described in the SD scheme.

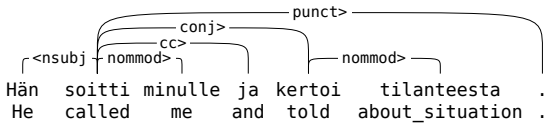

Figure 1: The basic variant of the Stanford Dependencies scheme on a Finnish sentence. The example can be translated as *He called me and told me about the situation*.

In addition to the post-processing approach implemented in the Stanford tools, also methods to directly parse dependency graphs involving tokens with multiple governors have been studied. McDonald and Pereira (2006) introduce a modification of the Maximum Spanning Tree algorithm to infer secondary dependencies in the Danish Dependency Treebank, and Sagae and Tsujii (2008) present a modification of the Shift-Reduce algorithm, which can parse directed acyclic graphs.

3 Data

3.1 Turku Dependency Treebank

As both the training and testing data of this study, we use the Turku Dependency Treebank (TDT) (Haverinen et al., 2011), which is a publicly available treebank for Finnish. TDT contains 15,126 sentences (204,399 tokens) from ten different genres or text sources, including for instance Wikipedia, EU-text and amateur fiction.

TDT has been annotated using the SD scheme, which was originally developed to be used with the English language. Thus it has been slightly modified in order to be able to capture the specific features of Finnish. The Finnish-specific SD scheme contains 53 different dependency types, and has been thoroughly described in the annotation manual of Haverinen (2012).

3.2 Conjunct propagation and additional dependencies

TDT contains two annotation layers. The first layer is based on the *basic* variant of the SD scheme and represents the structure of a sentence as a tree. Figure 1 illustrates the first annotation layer. The second layer gives extra information about specific phenomena: *conjunct propagation*,

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

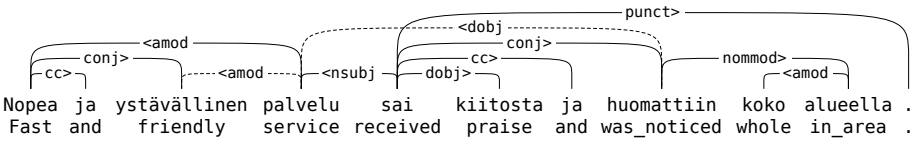

Figure 2: Conjunct propagation in the SD scheme. The base-layer dependencies are marked with solid lines, and the propagated dependencies are dashed. The example sentence can be translated as *The fast and friendly service received praise and was noticed in the whole area*. Note that the noun *palvelu* (*service*) serves as the subject of the first clause and the object of the second, which causes the type of the propagated dependency to change.

external subjects and *syntactic functions of relativizers*. Approximately 9% (18,926/208,417) of all dependencies in the treebank are part of the second layer. The second annotation layer adds dependencies on top of the existing first layer, thus making the resulting analyses directed graphs, rather than trees.

Conjunct propagation is related to coordination structures. In the SD scheme, the first coordinated element is considered to be the head of the whole coordination, and all other coordinated elements depend on it. Therefore, if a sentence element depends on the first element of a coordination, it can alternatively modify only the first element, some coordinated elements, or all of them. If a sentence element modifies multiple conjuncts, it should be propagated to them, as illustrated in Figure 2. Similarly, some or all conjuncts can modify another sentence element. If a modifier serves a different role for different conjuncts, or if coordinated elements are of different parts-of-speech, the type of the propagating dependency may change during the propagation. This is also illustrated in Figure 2.

External subjects occur with so called *open clausal complements*, where two verbs share a subject (also known as *subject control*). Due to the treeness restriction, in the basic layer of annotation it is not possible to convey the information that the subject of the first verb is also the subject of the second verb. Therefore, these subjects are marked in the second annotation layer, using the dependency types *xsubj*, for external subjects, and *xsubj-cop*, for external copula-subjects.

Syntactic functions of relativizers give additional information about relative clauses. The phrase containing the relative pronoun is marked simply as a *relativizer* (*rel*) in the first layer of annotation. However, the relativizer also always has a secondary syntactic function; for instance, it can

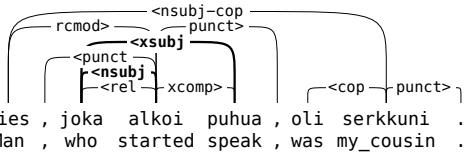

Figure 3: Syntactic functions of relativizers and external subjects. The relative pronoun *joka* (*who*) also acts as the subject to the main verb of the relative clause, as well as the subject of its open clausal complement. The example sentence can be translated as *The man who started to speak was my cousin*.

be the subject of the relative clause. This is marked in the second annotation layer with an additional dependency, which takes one of the dependency types defined in the first annotation layer. Due to the fact that the governor of the relativizer dependency is always the main predicate of the relative clause, the second layer dependency does not necessarily have the same governor. Both external subjects and relativizers are illustrated in Figure 3.

External subjects and syntactic functions of relativizers also interact with conjunct propagation. External subjects can propagate, and propagated subjects can produce new external subjects. Relativizers can also propagate, but note that if the relativizer dependency and the corresponding second layer dependency are between the same tokens, they always propagate together. Finally, if a relativizer acts as the subject of a predicate, it can also act as the external subject of another predicate.

4 Methods

We now proceed to describe the method that automatically infers the propagated and other additional dependencies based on the first layer of syntax annotation in the treebank. We have divided

the task into three different subproblems: conjunct propagation, syntactic functions of relativizers, and external subjects. The first two are solved using machine learning, whereas the third problem is easily approached with a rule-based method.

4.1 Conjunct propagation

In conjunct propagation, as is the case of the other two tasks as well, it is possible to exhaustively enumerate all candidate governor–dependent pairs between which may exist a dependency resulting from conjunct propagation. This is the case also for recursive coordination structures, in which the dependencies propagate along chains of two or more conjunct dependencies. We can therefore cast the problem as a multi-class classification task, whereby each of the candidate governor-dependent pairs is assigned the type of the propagated dependency, or alternatively classified as a negative example in case the dependency does not propagate. A simple binary classification into dependencies that do or do not propagate does not suffice, as this would not account for the 2.3% of cases in which the type of the propagated dependency differs from the original base-layer dependency, as discussed earlier in Section 3.2. In preliminary experiments, we have also tested a combined approach of binary classification (propagate or not) followed by a multiclass classification (assign type to propagated dependencies), but found that such a combined approach gives no additional advantage.

The set of possible classes consists of 49 dependency types from the SD scheme (for four SD types, *punct*, *conj*, *cc* and *ellipsis*, propagation is not allowed), plus the negative class and a number of compound types for relativizers. As mentioned in Section 3.2, the relativizer and its corresponding second layer dependency propagate together. Due to this, we have performed the propagation in two steps. First, the two dependency types are merged into one compound type, such as *rel-nsubj*, and after the propagation, they are separated again into two distinct dependencies. This merging increases the number of classes by ten, as only functions of relativizers that actually occur in the training data are allowed. Discounting dependency types that in fact never propagate in the training data and are thus never predicted to do so, the total number of possible classes is 51.

A number of features, extracted from both the tokens and the underlying dependency structure, are used in the classification. *Token features* include the lemma of the token, its main POS, and a separate feature for each its morphological tags that belong to one of several relevant morphological categories. These are *Subcategory*, *Case*, *Number*, *Person*, *Voice* and *Infinitive*, as selected based on preliminary experiments. Token features are extracted separately for the candidate governor and dependent, as well as the head of the coordination. The lemma of the coordinating conjunction itself, if such a conjunction is present, is also used as a feature. *Tree features* include the type of the dependency which is being propagated, whether the dependency governs or modifies the head of the conjunction, whether the target of the propagation already has a dependent with the same type (only relevant in cases where a dependent is propagated, not the governor), the set of outgoing dependency types for the candidate governor and dependent, the dependency type governing the head of the dependency being propagated, whether the linear direction of the candidate propagated dependency is the same as the linear direction of the dependency being propagated (possible values being both-left, both-right, and differing-directions), and finally the number of coordinated items in the coordination expressed as a binary feature (i.e. one binary feature for every discrete value).

Prior to classification, all possible feature pairs are explicitly generated, simulating the use of a second-degree polynomial kernel. For instance, a feature vector (f_1, f_2, f_3) is turned into $(f_1f_1, f_1f_2, f_1f_3, f_2f_2, f_2f_3, f_3f_3)$ prior to classification. As will be shown in the feature ablation study in Section 5.1, this technique improves the classification accuracy. Finally, prior to the classification the feature vectors are normalized to unit length.

4.2 Syntactic functions of relativizers

As discussed in Section 3.2, every relativizer is assigned a syntactic function in the second annotation layer of the treebank. This is expressed as an additional dependency that governs the relativizer (see Figure 3) and in the majority of cases (95.4%) has the same governor as the relativizer dependency in the first annotation layer. As with conjunct propagation, we approach the task as a machine learning problem. For each relativizer dependency, we predict an additional dependency

type which represents the syntactic function of the relativizer. A new dependency with this type is then created governing the relativizer word.

To account for certain cases of object control and raising, we identify cases where the governor of the relativizer has an infinite clausal complement (i.e. governs an *iccomp* dependency) and the head of the complement does not already have a dependent of the predicted type. If so, we post-process the new dependency to be governed by the head of the complement. This is a heuristic to treat the most common situation where the governor of the *relativizer* dependency differs from the governor of the corresponding second layer dependency. In all other cases, the second layer dependency is predicted between the governor and dependent of the base syntax relativizer dependency.

The feature representation in this task is comparatively simple. Separately for the governor and dependent, we generate their token features (the same features as in conjunct propagation) and the set of types of dependencies they govern. As with conjunct propagation, explicit feature pair generation as well as normalization of feature vectors to unit length are employed.

4.3 External subject assignment

Unlike in the two previous tasks, we find that assignment of external subjects is best approached by a simple rule-based method, since only clausal complements governed by an *xcomp* dependency have an external subject. As noted by Campbell (2004), in some highly restricted linguistic problems rule-based approaches are sufficient. The rule assigning external subjects only needs to account for whether the external subject dependency type is the regular subject type *xsubj* or the copular subject type *xsubj-cop*. Further, chains of clausal complements with external subjects must be correctly addressed, so that each open clausal complement correctly receives an external subject.

4.4 Combining predictions

As mentioned earlier in Section 3.2, the three tasks are not independent of each other: First, if the predicted syntactic function of the relativizer is a subject, the newly inserted subject dependency can produce an external subject dependency as well. Second, both external subjects and the dependencies encoding relativizer functions may propagate in coordinations. Third, propagated subject dependencies may again produce new external subject dependencies.

	P	R	F
Conj. propagation	93.1	92.9	93.0
Relativizer prediction	94.5	92.4	93.5
External subjects	90.0	97.3	93.5
All tasks combined	93.0	93.2	93.1

Table 1: Performance of the combined second layer prediction, as well as the individual tasks measured in terms of precision, recall, and F_1-score on the gold-standard base syntax trees.

The combined prediction of the entire second annotation layer is thus carried out in four separate steps. First, we predict the syntactic functions of relativizers, then the external subjects. After this, the conjuncts are propagated, and finally, the external subjects are predicted again, in order to cover external subjects produced by propagated subject dependencies.

4.5 Machine learning method and parameter selection

The underlying classifier for both the conjunct propagation and the relativizer syntactic function prediction is the multi-class support vector machine implemented in the *SVM-multiclass* package of Joachims (1999). The fast training algorithm implemented for linear kernels in SVM-multiclass is also the reason why we explicitly generate feature pairs, instead of directly utilizing the quadratic kernel. The available data is divided into training (80%), parameter optimization (10%), and test (10%) sets, this division being constant in all reported results. Further, the division is done on document level, i.e., all sentences from a single document in the treebank are assigned to the same set. This is to avoid any possibility of sharing information about the behavior of rare lexical items between the training and test sets. All reported results are obtained by optimizing the SVM regularization parameter C on the parameter optimization set, and using the resulting model on the test set. This optimization is done separately for each task.

5 Evaluation

We evaluate the performance of the predictions in terms of precision (P), recall (R), and F_1-score (F) of the predicted second layer dependencies. Precision is defined as the proportion of dependen-

	P	R	F
Full method	93.3	93.0	93.1
- feature pairs	92.5	92.1	92.3
- lemma	92.5	92.2	92.3
- lemma & morph	90.7	93.4	92.1
- lemma & morph & POS	87.9	91.9	89.9

Table 2: Feature ablation study. *Feature pairs* refer to the second-degree polynomial expansion described in Section 4.1, and *morph* refers to features extracted from morphological tags other than the main POS.

Method	P	R	F
Always	48.5	97.4	64.8
Type	61.8	51.6	56.2
Type and direction	83.5	64.1	72.6
Stanford parser alg.	83.7	57.7	68.3
Proposed method	93.3	93.0	93.1

Table 3: Performance of the proposed machine learning method in terms of precision, recall and F_1-score of propagated dependencies. The performance is compared to the four baselines defined in Section 5.1.

cies in the evaluated output also present in the gold standard, and recall as the proportion of dependencies in the gold standard also present in the evaluated output. Using these, F_1-score is defined as $F = \frac{2PR}{P+R}$. In addition to evaluating the performance using the gold-standard base layer annotation in the treebank, we also perform an evaluation with the base syntax layer produced by a dependency parser, discussed further in Section 5.2.

5.1 Performance and baselines

The evaluation results of the combined second layer prediction are shown in Table 1. The performance on the gold standard base syntax is high, with an overall F_1-score of 93.1%.

For conjunct propagation, which is the largest (71.4% of all second layer dependencies in the treebank are propagated) and arguably most important subtask, we perform several further analyses. Using the gold-standard syntactic information, also including gold-standard relativizer functions and external subjects, we estimate the contribution of the various feature types to the classification performance of the conjunct propagation subtask in a feature ablation study. The results of this study are shown in Table 2. Interestingly, an F_1-score of 89.9%, only 3.2pp lower than the full method, can be achieved only based on the features extracted from the syntactic tree, with no token-derived information whatsoever. Further, we see that using explicit feature pair generation improves the results by 0.8pp.

Next, we compare the performance of the machine learning conjunct propagation method to several baselines. The trivial baseline is to *always propagate*. We also implement a *propagate type* baseline, in which a dependency is propagated only if its type is more likely to propagate than not in the training data, regardless of whether the propagated dependency governs the head of the coordination, or depends on it. Taking into account the fact that dependencies governing the head of the coordination are considerably more likely to propagate (96.5% propagate) compared to those modifying it (32.9% propagate), in the *propagate type and direction* baseline a dependency is propagated only if dependencies with the same type and direction (i.e. govern or depend on the head of the coordination) are more likely to propagate in the training data.

As the primary baseline, we implement a close approximation of the conjunct propagation method in the Stanford Parser,[2] the "reference standard" for the SD scheme. The Stanford Parser conjunct propagation algorithm is relatively conservative, aiming at high precision at the cost of recall. All dependencies governing the head of the coordination are propagated, unless involved in a complex coordination of two relative clauses. Only subject dependencies governed by the head of the coordination are propagated, unless the propagation target already has a subject of its own. The type of a propagated subject dependency may change to/from the passive subject, depending whether the target of the propagation is active or passive. Our implementation differs in the handling of propagation in passive structures, since Finnish does not have passive subjects but rather direct objects.

The performance of the proposed method as well as the four abovementioned baselines is summarized in Table 3. In terms of F_1-score, the proposed method outperforms all baselines by a wide margin. Of particular interest is the gain over the algorithm used in the Stanford parser, the current

[2] `http://nlp.stanford.edu/software/lex-parser.shtml`, version 2.0.4

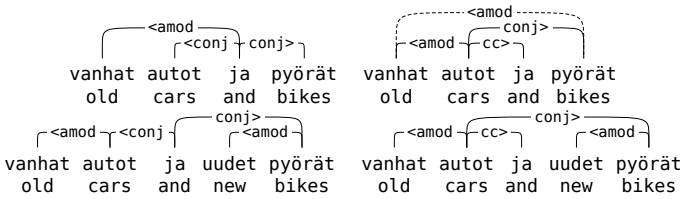

Figure 4: Left: the conjunction-as-head analysis, akin to the Link Grammar scheme. Modifiers of all coordinated elements are attached to the conjunction, while modifiers of a single coordinated element are attached to the element itself. In the top analysis, the adjective *vanhat* (*old*) modifies the whole coordination, and in the bottom analysis, only the first conjunct. Right: the corresponding analyses in the SD scheme. The example can be translated as *the old cars and (the new) bikes*.

widely-used reference implementation of conjunction propagation in the SD scheme.

5.2 Performance on parser output

Next, we discuss the performance of the proposed method on input produced by a dependency parser, as opposed to gold standard syntactic trees. This will also allow us to test one additional baseline, the conjunction-as-head analysis, discussed later in this section.

The parser used in the evaluation is a combination of the HunPOS tagger (Halácsy et al., 2007) with the Mate-Tools statistical dependency parser of Bohnet (2010), a second-order graph-based parser that achieves a state-of-the-art performance on a number of different languages; an earlier version of this parser ranked first on English and German in the CoNLL shared task in 2009 (Hajič and others, 2009). With a labeled attachment score of 81%, this combination represents the best dependency parser currently available for Finnish as tested on the Turku Dependency Treebank, outperforming for instance the popular MaltParser (Nivre et al., 2007) by several percentage points.[3]

When parser output is considered, a drop in performance is to be expected, seeing that coordination is one of the hardest phenomena to parse, and the parser often fails to produce the dependencies needed in order to generate the propagated dependencies. The evaluation on top of the parser output is presented in Table 4. The overall F_1-score is 61.8% (compared to 93.1% on gold standard syntax), and on the coordination propagation task only, the F_1 score is 58.4% (compared to 93.0% on gold standard syntax). This performance drop

[3]A detailed description of the parser pipeline is out-of-scope for this paper. The parser is described in a manuscript currently under review.

	P	R	F
Conj. propagation	58.1	58.6	58.4
Relativizer prediction	85.5	83.3	84.4
External subjects	67.5	73.0	70.1
All tasks combined	61.3	62.2	61.8

Table 4: Performance of the combined second layer prediction, as well as the individual tasks measured in terms of precision, recall, and F_1-score on top of statistical parser output.

can be attributed to the accuracy of the underlying parse trees, since the correct base syntax structure is present only for 66.1% of the propagated dependencies in the gold standard, thus imposing a severe restriction on the recall of the conjunct propagation method. This reflects the intrinsic difficulty of syntactically parsing coordination structures. In contrast, 89.1% of relativizer dependencies are correctly recovered by the base parser, allowing a much higher recall on this task. Out of all errors in all three subtasks, approximately 79.2% can be attributed to the parser output not containing the required base structure, meaning that in fact, the performance of the machine learning method itself does not degrade notably when applied to parser output.

We also repeat the baseline experiments discussed above using parser output rather than gold standard dependencies. Since reliable external subject and relativizer function dependencies are not available for the parser output, we disregard these. The results are given in Table 5, demonstrating that the performance of the proposed method is clearly superior to all of the baselines also on parser output.

As a final point of comparison, we test a joint approach to parsing and conjunct propaga-

Method	P	R	F
Always	30.5	62.3	40.9
Type	37.9	29.8	33.4
Type and direction	50.9	38.2	43.6
Stanford parser alg.	54.1	38.5	44.9
Proposed method	57.3	58.0	57.7

Table 5: Performance of the method as compared to the baselines of Section 5.1 on top of parser output.

tion, by adopting an analysis where the conjunction is the head of the coordination structure, as has been done for instance in the Link Grammar parser (Sleator and Temperley, 1993). In this scheme, a dependent modifying a single coordinated element is governed by this element whereas a dependent modifying all of the coordinated elements is governed by the coordinating conjunction. This approach is illustrated in Figure 4. The most important property of this representation is that in both cases, the resulting analysis is a tree, which in turn can be used to train a dependency parser, thus combining base syntax parsing with conjunct propagation as a single joint task.

We have developed a forward and backward conversion to the conjunction-as-head style. Note that this conversion is not lossless, as there are several structures which this analysis cannot express: dependents modifying multiple but not all coordinated elements, and cases where the governor to the head of the coordination does not propagate. Further, dependencies whose type changes as a result of the propagation cannot be represented either. A difficulty is also presented by cases where no explicit conjunction is stated in the text, nor is there a punctuation symbol (such as a comma) which would serve its role. These cases, however, only occur in 0.5% of all sentences, which we subsequently discard. Applying the forward and backward conversion to the gold-standard data results in a precision of 92.6% and a recall of 92.3% in propagated dependencies, demonstrating that the majority of cases are within scope for the conjunction-as-head analysis. Finally, note that this style cannot directly represent the other second layer dependencies like external subjects.

Using again the Mate-Tools parser of Bohnet (2010), trained on the treebank transformed in the conjunction-as-head style, the performance on propagated base layer dependencies is shown in Table 6 and compared to the proposed machine learning method, re-trained to match the input data (i.e. no external subjects or relativizer syntactic functions present). Here we see that the joint parsing and propagation perform notably worse in comparison with the proposed method. This agrees with the results of Schwartz et al. (2012), who show in their studies about learnability of different syntactic schemes that making the first conjunct of coordination as a head improves parsing results significantly. It is also important to remember that the conjunction-as-head analysis incurs a notable penalty for not being able to represent approximately 7% of the conjunct propagation cases in the data, as demonstrated by the recall of the forward and backward conversion.

	P	R	F
Conj. as head	43.7	42.6	43.1
Proposed method	58.2	58.8	58.5

Table 6: Comparisons of results obtained by reverse converting to SD the output of a statistical parser trained to produce the conjunction-as-head style of analysis, with the proposed method.

5.3 Discussion

When examining the results presented in this paper, two issues should be noted. First, although the conjunct propagation of the SD scheme is indeed closely related to the resolution of coordination scope ambiguity, it is not the entirety of this difficult disambiguation problem. Consider, for instance, the English phrase *corn and peanut butter*. This phrase contains a coordination ambiguity: either it describes butter made of corn and peanuts, or one of the items described is corn and the other peanut butter. However, this ambiguity lies deeper than the conjunct propagation layer of the SD scheme, as illustrated in Figure 5. As a result, when the method presented in this paper is applied, this particular ambiguity has already been resolved by the parser that has produced the base-syntactic trees.

Second, a similar note applies to the specific nature of the Finnish language, or the Finnish compound nouns in particular. Unlike for instance in English, in Finnish it is customary to write compounds as one word. As a consequence, this particular ambiguity is not very problem-

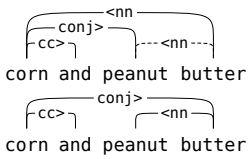

Figure 5: The ambiguity of the phrase *corn and peanut butter* is to be resolved in the *basic* variant of the SD scheme, not in the conjunct propagation layer. Top: the reading where the butter is made of corn and peanuts. Bottom: the reading where corn is combined with peanut butter. Note that while there is a propagated dependency present in the top reading, the decision whether this dependency should be generated or not does not represent the ambiguity of the phrase, and in fact, there is no valid reading where the propagated dependency would be absent, that is, a reading where the butter would be made of corn but not peanuts.

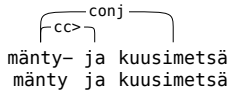

Figure 6: The difference between a simple noun coordinated with a compound and a two-part compound is surface-marked in Finnish using a dash. The top phrase can be translated as *a pine and fir forest* and the bottom phrase as *a pine and a fir forest*.

atic in Finnish, since the difference is surface-marked using a dash. For instance, the coordination *mänty- ja kuusimetsä* (*a pine and fir forest*) describes a forest growing both pines and firs, whereas *mänty ja kuusimetsä* (*a pine and a fir forest*) is the coordination of a single pine combined with a fir forest. Also, as breaking compound words into their components during the syntax annotation is not allowed, the analyses of the two Finnish phrases in TDT would in fact be identical, as illustrated in Figure 6, and would not involve propagation of dependencies.

6 Conclusion

In this paper, we have introduced a method for inferring additional sentence structure information from a dependency parse tree in the Stanford Dependencies scheme, most importantly *propagation of conjunct dependencies*, which is related to resolving *coordination scope ambiguities*. This machine learning based method uses the syntactic trees and morphological information to predict the additional dependencies, which can be highly useful in for instance the construction of a PropBank, as previously demonstrated by Haverinen et al. (2010).

On gold standard syntactic trees, the method achieves 93.1% F_1-score. When evaluated on top of actual parser output rather than gold standard trees, the performance predictably suffers a penalty, but an analysis of the errors reveals that 79.2% of all errors, when evaluated on parser output, are due to the parser not producing the correct base structure and thus disallowing the method from retrieving the correct dependencies.

In addition, we have also separately evaluated the largest and most important subtask, the conjunct propagation by comparing it against several baseline methods, including the method used in the original Stanford tools. We find that the proposed method clearly outperforms all baselines, and in particular, it achieves improved results over the method used in the original Stanford tools, which are widely used for producing the additional dependencies in applications, and can thus serve as its more accurate replacement in all applications that rely on syntactic analysis in the SD scheme. Interestingly, we also find that when using no token-based information, an F_1-score of 89.9% can be achieved for this subtask, only 3.2pp lower than the full, lexicalized set of features. This demonstrates that much of the necessary information for the task is contained in the syntactic trees themselves.

The software used in this work will be integrated with the existing Finnish parser and made publicly available at the address `http://bionlp.utu.fi/`, under an open licence. The training data will be available for the public in the final version of the Turku Dependency Treebank.

Acknowledgments

This work was supported by the Academy of Finland and the Emil Aaltonen Foundation. Computational resources were provided by CSC – IT Center for Science.

References

Rajeev Agarwal and Lois Boggess. 1992. A simple but useful approach to conjunct identification. In *Proceedings of ACL'92*, pages 15–21.

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

Joan Bresnan. 2001. *Lexical-functional syntax*, volume 16 of *Blackwell Textbooks in Linguistics*. Blackwell.

Richard Campbell. 2004. Using linguistic principles to recover empty categories. In *Proceedings of ACL*, pages 646–653.

Francis Chantree, Adam Kilgarriff, Anne de Roeck, and Alistair Willis. 2005. Disambiguating coordinations using word distribution information. In *Proceedings of RANLP'05*.

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

Miriam Goldberg. 1999. An unsupervised model for statistically determining coordinate phrase attachment. In *Proceedings of ACL'99*, pages 610–614.

Jan Hajič et al. 2009. The conll-2009 shared task: syntactic and semantic dependencies in multiple languages. In *Proceedings of CoNLL'09*, pages 1–18.

Péter Halácsy, András Kornai, and Csaba Oravecz. 2007. HunPos – an open source trigram tagger. In *Proceedings of ACL'07, Companion Volume*, pages 209–212.

Katri Haverinen, Filip Ginter, Veronika Laippala, Timo Viljanen, and Tapio Salakoski. 2010. Dependency-based propbanking of clinical Finnish. In *Proceedings of LAW IV*, pages 137–141.

Katri Haverinen, Filip Ginter, Veronika Laippala, Samuel Kohonen, Timo Viljanen, Jenna Nyblom, and Tapio Salakoski. 2011. A dependency-based analysis of treebank annotation errors. In *Proceedings of Depling'11*, pages 115–124.

Katri Haverinen. 2012. Syntax annotation guidelines for the Turku Dependency Treebank. Technical Report 1034, Turku Centre for Computer Science, January.

Thorsten Joachims. 1999. Making large-scale SVM learning practical. In *Advances in Kernel Methods - Support Vector Learning*, pages 169–184. MIT Press.

Daisuke Kawahara and Sadao Kurohashi. 2007. Probabilistic coordination disambiguation in a fully-lexicalized Japanese parser. In *Proceedings of EMNLP-CoNLL'07*, pages 306–314.

Daisuke Kawahara and Sadao Kurohashi. 2011. Generative modeling of coordination by factoring parallelism and selectional preferences. In *Proceedings of IJCNLP'11*, pages 456–464.

Jin-Dong Kim, Sampo Pyysalo, Tomoko Ohta, Robert Bossy, Ngan Nguyen, and Jun'ichi Tsujii. 2011. Overview of bionlp shared task 2011. In *Proceedings of BioNLP Shared Task 2011 Workshop*, pages 1–6. Association for Computational Linguistics.

Tracy Holloway King, Crouch Richard, Stefan Riezler, Mary Dalrymple, and Ronald M. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of LINC'03*, pages 1–8.

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

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

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 81–88.

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

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

Philip Resnik. 1999. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of Artificial Intelligence Research*, 11:95–130.

Kenji Sagae and Jun'ichi Tsujii. 2008. Shift-reduce dependency dag parsing. In *Proceedings of the 22nd International Conference on Computational Linguistics (Coling 2008) - Volume 1*, pages 753–760.

Roy Schwartz, Omri Abend, and Ari Rappoport. 2012. Learnability-based syntactic annotation design. In *Proceedings of the 24th International Conference on Computational Linguistics (Coling 2012)*, pages 2405–2422.

Daniel Sleator and Davy Temperley. 1993. Parsing English with a Link Grammar. In *Proceedings of IWPT'93*, pages 277–291.

Gertjan van Noord. 2007. Using-self-trained bilexical preferences to improve disambiguation accuracy. In *Proceedings of IWPT'07*, pages 1–10.

Deniz Yuret, Laura Rimmell, and Aydin Han. 2012. Parser evaluation using textual entailments. *Language Resources and Evaluation*, pages 1–21.